文章编号：2096-1618(2021)05-0558-06

基于 Armijo 搜索步长的 BFGS 与 DFP 拟牛顿法的比较研究

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

摘要：拟牛顿法是求解无约束优化问题的重要方法，采用非精确 Armijo 准则确认搜索步长，其中初始点的选取采用两种不同的方法：利用 MATLAB 工具箱中的 rand 命令对 BFGS 和 DFP 两种算法的初始点进行随机选取；固定选择两个不同的初始点。讨论不同的初始点选取方法对两种算法收敛效率及结果的影响，最后对两种算法收敛效果进行比较研究。结果表明：在多项式函数中，初始点的选取方法对 DFP 法的收敛效率有一定影响，在低次函数中，DFP 法收敛效率更好，在高次函数中，使用 BFGS 法的收敛效果更好；在非多项式函数中，随机取点对计算结果有一定影响，选择离极小点近的点作为初始点得到的最小值更好，并且使用 BFGS 法的收敛速度更快。

关 键 词：无约束最优化；BFGS 拟牛顿法；DFP 拟牛顿法；Armijo 搜索
中图分类号：O221.2 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2021.05.014

0 引言

20 世纪 50 年代中期，美国物理学家 Davidon 首次提出拟牛顿法。不久后，这一算法被运筹学家 Fletcher 和 Powell 证明是当时既快又稳定的算法[1]。拟牛顿法避免了在计算过程中求解 Hesse 矩阵，其中避免了：由 Hesse 不正定得出的方向不是下降方向的问题；在大规模问题中，计算 Hesse 矩阵计算量过大的问题。拟牛顿法属于一种特殊的共轭梯度法，其主要思想是利用相邻两迭代点之间的位移和它们对应目标函数的梯度差构造 Hesse 矩阵的一个近似，然后基于牛顿方程产生搜索方向，继而通过线搜索完成迭代过程[2]。

拟牛顿法还有许多具体的方法，有 BFGS 拟牛顿法、DFP 拟牛顿法、秩 1 拟牛顿法等。其中，DFP 拟牛顿法在 1959 年由 Davidon 提出，后又由 Fletcher 和 Powell 进行改进和完善，是最早的拟牛顿法，也是秩 2 矩阵的一种更新。20 世纪 70 年代，由 Broyden 在 1970 年研究得出 BFGS 算法，是目前最有效的拟牛顿校正[3]。随着两种算法的提出，有许多学者对两种算法进行了研究，例如：刘庆吉等[4]提出了在 DFP 算法中引入一个修正步骤，使其在没有凸性假设的情况下也收敛。张恒等[5]提出了一种新的 BFGS 方法，数值结果表明改进的方法具有优越性。

拟牛顿法因其自身的优点，在物理电波、传感技术、信号处理等方面被广泛研究[6-8]。但在实际应用中，采取哪种拟牛顿法收敛效果更好的研究很少。文中针对 BFGS 拟牛顿法和 DFP 拟牛顿法，讨论在以下两种类型的函数中，随机取初始点和选择固定初始点的情况下，研究不同拟牛顿法的收敛效果，以便在实际应用中，选择恰当的拟牛顿法来求解，达到效率最优化的目的。

收稿日期：2021-01-04
基金项目：国家自然科学基金资助项目(71672013)

1 预备知识

考虑如下无约束优化问题：

$$\min f(x) \tag{1}$$

$f(x)$是一个连续可微函数，其拟牛顿法的迭代公式为

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\lambda_k\boldsymbol{d}_k \tag{2}$$

$$\boldsymbol{d}^{(k)}=-\boldsymbol{H}_k\boldsymbol{g}_k \tag{3}$$

其中：$\boldsymbol{d}_k$ 为 $f(x)$ 在 $\boldsymbol{x}_k$ 处的搜索方向；$\boldsymbol{g}_k$ 为 $\boldsymbol{x}_k$ 处的梯度；$\boldsymbol{H}_k$ 为 Hesse 矩阵的近似；λ_k 为步长因子。

在式(2)中，$\boldsymbol{x}_k$ 沿着搜索方向 $\boldsymbol{d}_k$ 进行搜索，为达到 $\boldsymbol{x}_k$ 在这个搜索方向上能够搜索到最小值的目的，λ_k 应该满足下列条件

$$\lambda_k=\beta^{m_k} \tag{4}$$

其中，m_k 是满足下列不等式的最小非负整数 m，$\beta\in(0,1)$，$\sigma\in(0,0.5)$

$$f(\boldsymbol{x}_k+\beta^m\boldsymbol{d}_k)\leqslant f(\boldsymbol{x}_k)+\sigma\beta^m\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{d}_k \tag{5}$$

1.1　DFP 拟牛顿法

DFP 校正是第一个拟牛顿校正,为保证在采取 Armijo 准则时 $\boldsymbol{H}_k$ 的正定性能够满足,采取如下修正的校正公式:

$$\boldsymbol{H}_{k+1}=\begin{cases}\boldsymbol{H}_k & \boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{y}_k\leqslant 0\\ \boldsymbol{H}_k-\dfrac{\boldsymbol{H}_k\boldsymbol{y}_k\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{H}_k}{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{H}_k\boldsymbol{y}_k}+\dfrac{\boldsymbol{s}_k\boldsymbol{s}_k^{\mathrm{T}}}{\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{y}_k} & \boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k>0\end{cases}\tag{6}$$

其中:$\boldsymbol{H}_k$ 为 Hesse 矩阵的近似,$\boldsymbol{s}_k$ 为位移,$\boldsymbol{y}_k$ 为梯度差。

1.2　BFGS 拟牛顿法

BFGS 算法是区别于 DFP 的另一种秩 2 矩阵的更新,为保证在采取 Armijo 准则时 $\boldsymbol{B}_k$ 的正定性能够满足,采取如下修正的校正公式:

$$\boldsymbol{B}_{k+1}=\begin{cases}\boldsymbol{B}_k & \boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k\leqslant 0\\ \boldsymbol{B}_k-\dfrac{\boldsymbol{B}_k\boldsymbol{s}_k\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{B}_k}{\boldsymbol{s}_k^{\mathrm{T}}\boldsymbol{B}_k\boldsymbol{s}_k}+\dfrac{\boldsymbol{y}_k\boldsymbol{y}_k^{\mathrm{T}}}{\boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{y}_k} & \boldsymbol{y}_k^{\mathrm{T}}\boldsymbol{s}_k>0\end{cases}\tag{7}$$

其中,$\boldsymbol{B}_k$ 为 Hesse 矩阵的近似,$\boldsymbol{s}_k$ 为位移,$\boldsymbol{y}_k$ 为梯度差。

1.3　Armijo 搜索步长

在拟牛顿法中,为保证函数的收敛性,要求线搜索单调。大多数线搜索准则要求具有下降性质:$f(x_{k+1})<f(x_k)$,即所谓的单调搜索[9]。但精确线搜索因其计算量过大,会耗费许多的计算资源。特别地,在算法中对初始点进行随机选择,更加充满了不确定性,从而导致精确线搜索的结果不是有效且合理的。而 Armijo 非精确线搜索较为简单,易于实现其程序。所以,采取 Armijo 非精确线搜索方法,此方法在近几年也被广泛研究使用[9-13],Armijo 搜索准则公式已经在上述给出,这里不再列出。

1.4　拟牛顿法算法步骤

算法 1　(Armijo 搜索下的 BFGS 算法步骤)

Step1:选定参数 $\sigma\in(0,0.5)$,$\beta\in(0,1)$,根据实际情况设置允许误差 $0\leqslant\varepsilon\leqslant 1$,初始对称正定阵 $\boldsymbol{B}_0$(通常取为 $\boldsymbol{G}(x_0)$ 或单位阵 $\boldsymbol{I}_0$),令 $k=0$;

Step2:随机选择初始点 $\boldsymbol{x}_0$ 或给定初始点 $\boldsymbol{x}_0$;

Step3:计算 $\boldsymbol{g}_k$。若 $\|\boldsymbol{g}_k\|\leqslant\varepsilon$,则停止计算,输出 $\boldsymbol{x}_k$ 作为 $f(x)$ 极小点。否则,继续下一步。

Step4:解线性方程组 $\boldsymbol{B}_k\boldsymbol{d}=-\boldsymbol{g}_k$,求得解 $\boldsymbol{d}_k$;

Step5:由上述 Armijo 准则的数学表达式(5)确定步长因子 λ_k,其中 $\boldsymbol{x}_{k+1}$ 由式(2)给出;

Step6:由式(7)确定 $\boldsymbol{B}_{k+1}$;

Step7:令 $k=k+1$,转 Step1。

算法 2　(Armijo 搜索下的 DFP 算法步骤)

Step1:选定参数 $\sigma\in(0,0.5)$,$\beta\in(0,1)$,根据实际情况设置允许误差 $0\leqslant\varepsilon\leqslant 1$,初始对称正定阵 $\boldsymbol{H}_0$(通常取为 $\boldsymbol{G}(x_0)^{-1}$ 或单位阵 $\boldsymbol{I}_0$),令 $k=0$;

Step2:随机选择初始点 $\boldsymbol{x}_0$ 或给定初始点 $\boldsymbol{x}_0$;

Step3:计算 $\boldsymbol{g}_k$。若 $\|\boldsymbol{g}_k\|\leqslant\varepsilon$,则停止计算,输出 $\boldsymbol{x}_k$ 作为 $f(x)$ 极小点。否则,继续下一步。

Step4:解线性方程组 $\boldsymbol{d}_k=-\boldsymbol{H}_k\boldsymbol{g}_k$,求得解 $\boldsymbol{d}_k$;

Step5:由上述 Armijo 准则的数学表达式(5)确定步长因子 λ_k,其中 $\boldsymbol{x}_{k+1}$ 由式(2)给出;

Step6:由式(7)确定 $\boldsymbol{H}_{k+1}$;

Step7:令 $k=k+1$,转 Step1。

2　算法案例

此次计算基于 MATLAB($R2020a$),通过随机选择初始点或者固定选取不同的初始点,用 BFGS 法和 DFP 法对不同的函数进行迭代求最小值,并得到计算结果 x,迭代次数 k,函数的最小值 val,希望得到的目标函数值尽可能地接近于函数的最小值。考虑到容许误差取太小会造成迭代次数增多,设置容许误差为 10^{-6},若目标函数在此容许误差之下收敛步数过大、目标函数过于复杂,考虑容许误差为 10^{-4}。由于采取的是 Armijo 非精确线搜索准则确定步长,可能会出现计算结果达不到容许误差的要求,也为了使两种算法对比强烈、结果更为清晰明了,便于直观地分析两种算法的优缺点,把两种算法的最大迭代次数设置为 40000。

2.1　考虑多项式函数无约束优化问题

实例 1　$\min f(x)=x_1^2+x_2^2-2x_1-x_1x_2+2$

设置容许误差为 10^{-6},对于上述结构简单的二元二次函数,考虑到初始点对两种方法收敛性的影响,这里对初始点进行两种取法:一是利用 MATLAB 中的 rand 命令对随机数 1 ~ 100 进行随机取点,即随机取两个初始点 $(-4,-44)^{\mathrm{T}}$、$(32,96)^{\mathrm{T}}$,计算结果见表 1、表 2。二是通过计算得出的极小点,在极小点附近选择两个初始点进行迭代,即初始点为 $(2,4)^{\mathrm{T}}$、$(1,0)^{\mathrm{T}}$,计算得到的结果见表 3、表 4。

表 1　实例 1 初始点为 $(-4,-44)^{\mathrm{T}}$ 的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	9	$(1.3333,0.6667)^{\mathrm{T}}$	0.6667
DFP 法	5	$(1.3333,0.6667)^{\mathrm{T}}$	0.6667

表 2 实例 1 初始点为$(32,96)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	10	$(1.3333,0.6667)^T$	0.6667
DFP 法	5	$(1.3333,0.6667)^T$	0.6667

表 3 实例 1 初始点为$(2,4)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	8	$(1.3333,0.6667)^T$	0.6667
DFP 法	5	$(1.3333,0.6667)^T$	0.6667

表 4 实例 1 初始点为$(1,0)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	8	$(1.3333,0.6667)^T$	0.6667
DFP 法	4	$(1.3333,0.6667)^T$	0.6667

比较分析:表 1 ~4 说明 BFGS 法在不同的初始点下进行迭代,得到的迭代次数不同,但相差不大。而在表 3、表 4 中,BFGS 法因为初始点距离极小点近,迭代次数有所降低,这说明初始点的选取对 BFGS 法的效率还是有一定影响。无论是随机取的初始点,还是固定初始点,BFGS 法都能得到极小点以及最优解。表 1 ~3 得到 DFP 法的迭代次数都是 5,并且初始点$(2,4)^T$相较于随机取的两个初始点,其距离极小点很近,但 DFP 法得到的迭代次数仍然是 5 次。而在初始点为$(1,0)^T$的情况下,DFP 法的迭代次数变为 4,说明其迭代次数可以降低,但要求初始点越过某一临界点,此处对临界点不做讨论。在这个目标函数中,DFP 法明显地在收敛速度上优于 BFGS 法。

实例 2 $\min f(x)=100(x_2-x_1^2)^2+(1-x_1)^2$

设置容许误差为10^{-4},对于经典的 RosenBrock 函数,考虑到其为非凸函数,为更好测试两种算法的性能,利用 MATLAB 中的 rand 命令对随机数 1 ~10 进行随机取初始点 x_0,即随机取两个初始点$(10,7)^T$、$(-10,-7)^T$,计算得到的结果见表 5、表 6。为避免算法在最大迭代次数下达不到容许误差要求的极小值,选择两个离极小点较近的初始点进行迭代,即$(2,2)^T$、$(4,2)^T$,计算得到的结果见表 7、表 8。

表 5 实例 2 初始点为$(10,7)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	18	$(1.0000,1.0000)^T$	0
DFP 法	22	$(1.0000,1.0000)^T$	6.4038×10^{-13}

表 6 实例 2 初始点为$(-10,-7)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	19	$(1.0000,1.0000)^T$	6.6391×10^{-13}
DFP 法	7610	$(1.0000,1.0000)^T$	6.4038×10^{-13}

表 7 实例 2 初始点为$(2,2)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	15	$(1.0000,1.0000)^T$	0
DFP 法	36	$(1.0000,1.0000)^T$	2.1230×10^{-11}

表 8 实例 2 初始点为$(4,2)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	13	$(1.0000,1.0000)^T$	1.4211×10^{-14}
DFP 法	13	$(1.0000,1.0000)^T$	1.4222×10^{-13}

比较分析:从随机取点的表 5、表 6 得出 BFGS 法的迭代次数分别为 18、19,尤其在表 6 中,BFGS 法的迭代次数远远低于 DFP 法。无论随机取点还是固定点,两种算法都能得到精确解$(1,1)^T$。在表 8 中,虽然两种算法迭代次数一样,但通过 BFGS 法迭代得到的最优值更接近极小值。在表 6 中,DFP 法的迭代次数异常高,这说明最开始的步长没有使函数值下降到一个满意的值,从而为了求得一个满意的下降量,不停地对步长进行压缩,进而增加了迭代次数[2]。初始点的选取方法对两种算法结果影响不大,因为可以从表中看出最后的结果还是足够小,很接近 0,但对 DFP 法的收敛效率有一定影响。从表 5 ~8 可以看出,确定离精确解距离近的初始点,能对 DFP 法的迭代次数能够进行优化。但在表 8 中,虽然优化了 DFP 法的迭代次数,但其最后得到的函数值还是略差于 BFGS 法得到的函数值。这说明了在 RosenBrock 函数中,BFGS 法的性能比 DFP 法好。

实例 3 $\min f(x)=100(x_1+x_2-3)^2+(x_1-x_2+1)^4$

设置容许误差为10^{-4},对于上述高次函数,初始点的选取方法如下:一是利用 MATLAB 中的 rand 命令对随机数 1 ~10 进行随机取初始点,即随机选取两个初始点$(1,9)^T$、$(-8,-8)^T$,计算得到的结果见表 9、表 10;二是选择值相同的点为初始点,即$(10,10)^T$,再选择距离极小点较远的点为初始点,即$(80,90)^T$,计算得到的结果见表 11、表 12。

表 9 实例 3 初始点为$(1,9)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	20	$(0.9893,2.0107)^T$	2.1263×10^{-7}
DFP 法	20	$(0.9891,2.0109)^T$	2.2449×10^{-7}

表 10 实例 3 初始点为$(-8,-8)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	16	$(1.0100,1.9900)^T$	1.6076×10^{-7}
DFP 法	20	$(1.0055,1.9945)^T$	1.4835×10^{-8}

表 11　实例 3 初始点为$(10,10)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	16	$(1.0119,1.9881)^T$	3.2571×10^{-7}
DFP 法	17	$(1.0115,1.9885)^T$	2.7626×10^{-7}

表 12　实例 3 初始点为$(80,90)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	22	$(0.9897,2.0103)^T$	1.7714×10^{-7}
DFP 法	22	$(0.9897,2.0103)^T$	1.8007×10^{-7}

比较分析:在表 9 中,BFGS 法和 DFP 法的迭代次数都为 20,但是 BFGS 法的函数值更小。而在表 10 中,虽然 DFP 法的函数值更小,但 BFGS 法的迭代次数明显少于 DFP 法,这说明在随机取点的情况下,BFGS 法收敛效果更好。在表 11、表 12 中,BFGS 法的迭代次数低于 DFP 法一次,但是 BFGS 法得到的最小值略差;而在两种方法迭代次数一样的情况下,BFGS 法的结果更优。总的来看,初始点选取方法的改变不会影响算法效率以及最后的计算结果。并且在选择距离较远的值作为初始点的情况下,其迭代次数并没有增加多少,这说明初始点距离极小点的远近,对迭代次数影响不大。在上述高次函数中,选择 BFGS 法进行迭代收敛效果会更好。

通过上述 3 个不同的多项式函数实例,在实例 1 中,DFP 法收敛速度更快,初始点的选取方法对两种方法的收敛效率影响较小。但在实例 2 和实例 3 中,BFGS 法的收敛效率优于 DFP 法,并且初始点的选取对 DFP 方法迭代次数有较大影响,由文献[2]得知,若最初的函数下降量未达到一个满意值,则会对步长进行不断压缩,以求得一个满意的下降量,从而影响迭代效率。

2.2　考虑非多项式函数无约束优化问题

实例 4　$\min f(x)=2e^{-x_1}\sin(x_2)$

对于上述由指数函数和正弦函数相乘作成的函数,考虑到最大迭代次数的限制,以免得不到容许误差内的极小值,设置容许误差为 10^{-4}。并利用 MATLAB 中的 rand 命令对随机数 1 ~ 10 进行随机取初始点 x_0,即初始点取$(2,2)^T$、$(9,7)^T$,计算结果见表 13、表 14。再固定取两个初始点,即$(10,28)^T$、$(5,9)^T$,计算得到的结果见表 15、表 16。

表 13　实例 4 初始点为$(2,2)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	409	$(10.9613,7.5394)^T$	3.3018×10^{-5}
DFP 法	466	$(11.0055,7.5585)^T$	3.1782×10^{-5}

表 14　实例 4 初始点为$(9,7)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	7	$(11.0024,5.4901)^T$	-2.3745×10^{-5}
DFP 法	7	$(11.0024,5.4901)^T$	-2.3744×10^{-5}

表 15　实例 4 初始点为$(2,5)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	13	$(10.5374,3.6921)^T$	-2.7754×10^{-5}
DFP 法	13	$(10.5283,3.6568)^T$	-2.6378×10^{-5}

表 16　实例 4 初始点为$(5,9)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	12	$(10.4335,13.9141)^T$	5.7400×10^{-5}
DFP 法	12	$(10.4038,13.8897)^T$	5.8790×10^{-5}

比较分析:在表 13 ~ 16 中,两种方法得到的计算结果距离很近,其得到的最小值也很接近。但在表 14 ~ 16 中,两种方法得到的迭代次数相同,但 BFGS 法得到的最小值离极小值更近。在表 13 中,虽然 BFGS 法得到的最小值次于 DFP 法,但是 BFGS 法的迭代次数明显地要低于 DFP 法的迭代次数,这说明在此随机取点的情况下,得到的最小值可能会比 DFP 法差,但是随机取点不会改变两种算法在此函数中的效率;而固定选择的初始点,可以在一定范围内控制两种方法的迭代次数,不会出现表 13 中迭代次数极大的情况。综上,在这个有波动的复杂函数中,BFGS 法的收敛效率更好。

实例 5　$\min f(x)=e^{x_1}(2x_1^2+x_2^2+x_1x_2+x_2+2)$

设置容许误差为 10^{-4},对于上述结构复杂含指数函数的非多项式函数,考虑到最大迭代次数的限制,为避免得不到容许误差内的极小值,利用 MATLAB 中的 rand 命令对随机数 1 ~ 10 进行随机取初始点 x_0,即初始点取$(1,3)^T$、$(6,10)^T$,计算得到的结果见表 17、表 18。再固定选择两个初始点,即$(-2,-5)^T$、$(0,0)^T$,计算得到的结果见表 19、表 20。

表 17　实例 5 初始点为$(1,3)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	15	$(-0.7143,-0.1428)^T$	1.4686
DFP 法	16	$(-0.7145,-0.1427)^T$	1.4686

表 18　实例 5 初始点为$(6,10)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	33	$(-15.4189,0.4939)^T$	9.4696×10^{-5}
DFP 法	33	$(-15.5948,-0.1720)^T$	8.2848×10^{-5}

表 19 实例 5 初始点为$(-2,-5)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	15	$(-16.0797,-3.2626)^T$	6.0161×10^{-5}
DFP 法	15	$(-16.0787,-3.2520)^T$	6.0182×10^{-5}

表 20 实例 5 初始点为$(0,0)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	8	$(-0.7142,-0.1429)^T$	1.4686
DFP 法	9	$(-0.7144,-0.1429)^T$	1.4686

比较分析:从表 17 可以看出,两种算法在相同的点下得到的函数值一样,但 BFGS 法的迭代次数为 15,其收敛速度更快。在表 18、表 19 中,两种算法的迭代次数相同,但在最后得到的计算结果和函数值有一定的差距,可以看出 BFGS 法得到的结果更接近极小值,这说明 BFGS 法收敛速度更快。在表 17、表 20 中,虽然 DFP 法和 BFGS 法得到的最小值相等,但 DFP 法的迭代次数大于 BFGS 法的迭代次数。这说明无论初始点如何取值,DFP 法的性能效率略低。并且在随机选取初始点得到的结果与固定选取的初始点差别不大,这说明初始点的选取方法在此函数中对计算结果并无太大影响,而对算法迭代次数有一定影响。

实例 6 $\min f(x)=2x_1^2+2x_2^2+50\sin(\frac{\pi}{2}x_1)+50\cos(\frac{\pi}{2}x_2)+10$

对于上述结构复杂的二元函数,同样设置容许误差为 10^{-4}。同样地,利用 MATLAB 中的 rand 命令对随机数 1 ~ 10 进行随机取初始点 x_0,即$(2,8)^T$、$(1,3)^T$,计算得到的结果见表 21、表 22。考虑到该函数的波动性,选择一个距离极小点近的初始点,即$(-2,-2)^T$;再选择一个距离极小点远的初始点,即$(23,77)^T$,计算得到的结果见下表表 23、表 24。

表 21 实例 6 初始点为$(2,8)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	9	$(-2.2842,6.8430)^T$	17.5639
DFP 法	12	$(-2.2842,6.8430)^T$	17.5639

表 22 实例 6 初始点为$(1,3)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	16	$(-2.2842,6.8430)^T$	17.5639
DFP 法	1264	$(-2.2842,6.8430)^T$	17.5639

表 23 实例 6 初始点为$(-2,-2)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	7	$(-2.2842,6.8430)^T$	-68.4499
DFP 法	7	$(-2.2842,6.8430)^T$	-68.4499

表 24 实例 6 初始点为$(23,77)^T$的计算结果

计算方法	迭代次数 k	计算结果 x	最小值 val
BFGS 法	8	$(15.8025,6.8430)^T$	526.5530
DFP 法	9	$(15.8025,6.8430)^T$	526.5530

比较分析:从表 21 ~ 24 可以看出,4 个不同的初始点进行迭代,BFGS 法的迭代次数都比较低。在表 22 中,BFGS 法得到的函数值更接近极小值。而在随机选取初始点进行迭代的情况下,两种算法得到的最小值是优于表 24 得到的最小值,这说明随机取点会对计算结果产生影响。在表 23 中,选择距离极小点近的初始点$(-2,-2)^T$进行迭代,使得两种算法得到的结果一模一样。这说明在上述波动性较强的函数中,初始点的选取可以使两种算法收敛效果达到一样。在表 24 中,选择了距离极小点较远的初始点,DFP 法的迭代次数就要高于 BFGS 法迭代次数。但初始点选择离极小点较远的点,对最后的函数值是有一定影响。总体来看,BFGS 法和 DFP 法都能在最大迭代次数下达到容许误差内的最小值,但 BFGS 法收敛效果更好。综上,初始点的选取方法对算法的迭代速度以及计算结果有一定影响。

上述 3 个实例表明 BFGS 法的收敛效果更好。在实例 4、实例 5、实例 6 中,通过 rand 命令随机选择初始点,对拟牛顿法的迭代次数有很大影响。尤其是在实例 4 和实例 6 中,出现了 400 多次的迭代和 1264 次的迭代,这说明初始点的选择在此类函数中可能会对计算效率会有影响,并且选择离极小点近的点作为初始点会影响最后的最小值。

3 结束语

针对两种不同的拟牛顿法,通过对初始点选取方法的改变,初始点取值的不同以及算法不同,对无约束优化问题的求解进行 DFP 和 BFGS 比较分析。在多项式函数中,选择了 3 个具有代表性的例子测试算法性能,可以看出在低次多项式函数中 DFP 法的收敛效果较好;在高次函数中,BFGS 法收敛效率更好,而初始点的选取方法对函数最小值无太大影响,但随机选择的初始点可能会出现算法迭代次数过大的情况。在非多项式函数中,3 个实例都表明 BFGS 法的收敛效果更好,并且随机取点也无太大影响,但固定选择的初始点,可以减少迭代次数。为简化计算量以及便于程序实现,选择 Armijo 搜索,但 Armijo 准则最开始给出的试探步长对算法效率有一定影响,步长达不到满意的下降量,将导致函数迭代次数增加,因此 Armijo 准则

有利有弊,但对结果影响不大。总之,在低次多项式函数中,选择 DFP 法迭代效果更优,在高次多项式函数以及非多项式函数中,选择 BFGS 法效率更好,结果更优。文中为测试算法性能,初始点的选取有一定的主观性,在随机选择初始点的情况下规定了选择范围,在今后的学习中应该扩大初始点的选取范围。

参考文献:

[1] 马昌凤,柯艺芬,谢亚君. 最优化计算方法及其 MATLAB 程序实现[M]. 北京:国防工业出版社,2015.

[2] 王宜举. 非线性最优化理论与方法[M]. 北京:科学出版社,2012.

[3] Broyden C G. The convergence of a class of double-rank minimization algorithms[J]. J. Institute of Mathematics and Its Applications,1970,6(1):76-90.

[4] 刘庆吉,张长海. 一种修正的 DFP 方法[J]. 大庆石油学院学报,1990(1):100-104.

[5] 张恒,何伟. 基于新拟牛顿方程的改进 BFGS 方法[J]. 淮海工学院学报(自然科学版),2007,16(3):10-12.

[6] 潘和平,孟庆鑫. 地-井时域激电拟牛顿法反演问题研究[J]. 电波科学学报,2013,28(5):184-191.

[7] 徐耀松,谭亮. 基于改进拟牛顿-K 近邻的 TDOA 定位算法[J]. 传感技术学报,2018,31(10):122-127.

[8] 时伟. 基于新拟牛顿法的自适应均衡技术[J]. 北京邮电大学学报,2012,35(3):86-89.

[9] 张静. 关于线搜索的 Armijo 型方法[C]. 第十届中国青年信息与管理学者大会论文集. 2008.

[10] 张丽,周伟军. Armijo 线性搜索下 Hager-Zhang 共轭梯度法的全局收敛性[J]. 数学物理学报,2008,28(5):840-845.

[11] McCormick G P. A modification of Armijo´s step-size rule for negative curvature[J]. Mathematical Programming,1977,13(1):111-115.

[12] 黄飞,吴泽忠. 基于 Armijo 搜索步长的几种共轭梯度法的分析对比[J]. 成都信息工程大学学报,2019,34(2):209-215.

Comparison of BFGS and DFP Quasi-Newton Method based on Armijo Search Step

LI Juwen,　WU Zezhong

(College of Applied Mathematics, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: Quasi-Newton method is an important method to solve unconstrained optimization problems. In this paper, an inexact Armijo criterion is used to confirm the search step size, and two different methods are used to select the initial points : one is to use the rand command in MATLAB toolbox to randomly select the initial points of BFGS and DFP algorithms ; the other is to select two different initial points fixedly. The influence of different initial point selection methods on the convergence efficiency and results of the two algorithms is discussed. Finally, the convergence effect of the two algorithms is compared. The results show that in polynomial function, the selection method of initial point has certain influence on the convergence efficiency of DFP method. In lower order function, DFP method has better convergence efficiency. And in higher order function, BFGS method has better convergence effect. In the non-polynomial function, the random selection of points has a certain influence on the calculation results. It is better to select the point near the minimum point as the initial point, and the BFGS method has faster convergence speed.

Keywords: unconstrained optimization; BFGS Quasi-Newton method; DFP Quasi-Newton method ; Armijo search