文章编号：2096-1618(2024)03-0374-08

一类改进的拟牛顿算法

罗文军[1]， 吴泽忠[1]， 贺盛瑜[2]
(1. 成都信息工程大学应用数学学院，四川 成都 610225；2. 四川社会主义学院，四川 成都 610037)

摘要：在拟牛顿方程基础上，推导出一种新的 DFP 校正公式，并在强 Wolfe 步长规则下给出一类新的 DFP 算法。随后提出一种改进的强 Wolfe 线性搜索法，改善由于精度所导致的线性搜索失败的问题，并在一定假设下证明改进的算法具有全局收敛性。最后用算例来改进前后的 DFP 算法的性能作对比，结果表明改进的算法行之有效，并且具有更好的收敛性。

关 键 词：DFP 算法；共轭梯度；拟牛顿法；无约束最优化；线性搜索
中图分类号：O221.2 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2024.03.016

0 引言

拟牛顿算法是求解无约束优化问题最有效算法之一，提高了一阶方法的线性收敛速度，并实现局部超线性速率，而每次迭代的计算成本为 $O(d^2)$，而不是牛顿法的 $O(d^3)$。其利用目标函数值 f 和一阶导数 g 的信息构造出目标函数的曲率近似，经过多次迭代的近似 Hessian 矩阵，不需要形成 Hessian 矩阵。

拟牛顿算法有许多的校正公式，其中 DFP 校正公式是最早的拟牛顿校正公式[1]。Wei 等[2]在 2006 年提出了一种新的拟牛顿公式，在此拟牛顿公式的基础上，本文推导出了新的 DFP 校正公式，以及改进的线性搜索算法，并对算法进行收敛性分析。数值实验结果表明：改进的算法是可行的，并且与原始 DFP 算法相比，有更好的收敛性。

1 预备知识

考虑如下无约束优化问题：

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

式中 $f(x)$ 是二次可微函数，对其进行泰勒展开有：

$$f(x^{(k)}+s)\approx f(x^{(k)})+\nabla f(x^{(k)})^{\mathrm{T}}s+\frac{1}{2}s^{\mathrm{T}}\nabla^2 f(x^{(k)})s \tag{2}$$

式中 $s=(x-x^{(k)})$，对式(2)右边极小化有：

$$x^{(k+1)}=x^{(k)}-[\nabla^2 f(x^{(k)})]^{-1}\nabla f(x^{(k)}) \tag{3}$$

这就是牛顿法的迭代公式。

收稿日期：2022-11-29
基金项目：国家自然科学基金资助项目(71962030)；四川省社科重点研究基地资助项目(Xq21B06)；国家社会科学基金资助项目(21BTQ099)

为克服牛顿法中计算 Hessian 矩阵以及其逆矩阵时计算量大的缺陷，同时避免 Hessian 矩阵奇异或接近奇异的情况，一个直观的想法是通过不太复杂的过程构造目标函数 Hessian 矩阵的一个近似替换。这里的近似不是在数值上近似，而是在性能上近似。这就是拟牛顿算法的出发点[3]。

1.1 DFP 算法

对式(2)中的 x 求导得：

$$g(x)\approx g(x^{(k+1)})+\boldsymbol{G}(x^{(k+1)})(x-x^{(k+1)}) \tag{4}$$

式中，$g(x)=\nabla f(x)$ 为函数的一阶偏导，$\boldsymbol{G}(x)$ 为函数在 x 处的 Hessian 矩阵。令 $x=x^{(k)}$，$s_k=x^{(k+1)}-x^{(k)}$，$y_k=g(x^{(k+1)})-g(x^{(k)})$ 有：

$$g(x^{(k)})\approx g(x^{(k+1)})+\boldsymbol{G}(x^{(k)})(x^{(k)}-x^{(k+1)})$$

$$g(x^{(k+1)})-g(x^{(k)})\approx \boldsymbol{G}(x^{(k+1)})(x^{(k+1)}-x^{(k)})$$

$$y_k\approx \boldsymbol{G}(x^{(k+1)})s_k$$

$$\boldsymbol{G}_{k+1}^{-1}y_k\approx s_k \tag{5}$$

用 $\boldsymbol{H}_k$ 去近似 $\boldsymbol{G}_{k+1}^{-1}$，即 $\boldsymbol{H}_k\approx\boldsymbol{G}_{k+1}^{-1}$，便有拟牛顿方程，又称拟牛顿条件：

$$\boldsymbol{H}_k y_k=s_k$$

显然，即便要求 $\boldsymbol{H}_k$ 对称，拟牛顿方程中变量的个数也远大于方程的个数，因此拟牛顿方程的解不能唯一确定。对此，一个自然的想法就是通过对 $\boldsymbol{H}_k$ 进行某种修正得到 $\boldsymbol{H}_{k+1}$ 以求得拟牛顿方程的一个(组)特解，也就是：

$$\boldsymbol{H}_{k+1}=\boldsymbol{H}_k+\Delta\boldsymbol{H}_k$$

式中 $\Delta\boldsymbol{H}_k$ 成为修正项。Davidon，Fletcher 和 Powell 提出如下公式进行校正：

$$\boldsymbol{H}_{k+1}=\boldsymbol{H}_k+as_k s_k^{\mathrm{T}}+b\boldsymbol{H}_k y_k y_k^{\mathrm{T}}\boldsymbol{H}_k$$

式中 a 和 b 为待定常数。由拟牛顿条件知道：

$$\boldsymbol{H}_k y_k + a(s_k^{\mathrm{T}} y_k) s_k + b(y_k^{\mathrm{T}} \boldsymbol{H}_k y_k) \boldsymbol{H}_k y_k = s_k$$

设 s_k 与 $\boldsymbol{H}_k y_k$ 线性无关,则

$$a = \frac{1}{s_k^{\mathrm{T}} y_k}, \quad b = -\frac{1}{y_k^{\mathrm{T}} \boldsymbol{H}_k y_k}$$

这样就得到了 DFP 校正公式:

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \frac{s_k s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k} - \frac{\boldsymbol{H}_k y_k y_k^{\mathrm{T}} \boldsymbol{H}_k}{y_k^{\mathrm{T}} \boldsymbol{H}_k y_k} \tag{6}$$

使用 DFP 校正公式更新的拟牛顿算法叫作 DFP 算法,DFP 算法步骤如下:

步骤 1　设置初始点 $x^{(0)}$,$\boldsymbol{H}_k = I$;

步骤 2　计算 g_k,若 $\|g_k\| < \varepsilon$ 则停止;

步骤 3　令下降方向 $d_k = -\boldsymbol{H}_k g_k$;

步骤 4　沿方向 d_k 作线性搜索得到步长 α_k,并更新 $x^{(k+1)} = x^{(k)} + a^{(k)} d_k$;

步骤 5　用式(6)校正 $\boldsymbol{H}_k$ 产生 $\boldsymbol{H}_{k+1}$,转步骤 2。

1.2　线性搜索

线性搜索是拟牛顿法中至关重要的一步,目的是解决一个无约束最优化问题:

$$\min f(x^{(k)} + a^{(k)} d_k) \tag{7}$$

步长 α 的选取得当与否,能决定算法的收敛速度以及是否收敛。但是与每次迭代都进行精确线性搜索相比,非精确线性搜索更具有效率。

Armijo 条件:

$$f(x^{(k)} + \alpha_k d_k) \leqslant f(x^{(k)}) + \beta\alpha \nabla f(x^{(k)})^{\mathrm{T}} d_k \tag{8}$$

条件(8)也被称为充分下降条件,其中 $\beta \in [0,1]$ 通常令 $\beta = 10^{-4}$。其中 $\beta = 0$ 时,那么任何下降都是可以接受的,如果 $\beta = 1$,则下降必须至少与一阶近似预测值相同[4]。当 d 为有效的下降方向时,一定可以找到一个足够小的步长使之满足充分下降条件。因此可以从较大的步长进行缩小,直到满足充分下降条件为止,这种方法被称为回溯线搜索[5]。

但是条件(8)不能保证收敛到局部最小值,因此又有了另一个曲率条件:

$$\nabla f(x^{(k)} + a_k d_k)^{\mathrm{T}} d_k \geqslant \sigma \nabla f(x^{(k)})^{\mathrm{T}} d_k \tag{9}$$

式中 σ 控制下一个方向的深度,要求下一次迭代时方向倒数的曲率更小。

条件(8)和(9)共同组成了 Wolfe 条件,通常设置 $0 < \beta < \sigma < 1$,设置 $\sigma = 0.9$。

曲率条件还有一个更强限制的版本:

$$|\nabla f(x^{(k)} + a_k d_k)^{\mathrm{T}} d_k| \leqslant \sigma |\nabla f(x^{(k)})^{\mathrm{T}} d_k| \tag{10}$$

条件(8)和(10)共同组成了强 Wolfe 条件。

满足强 Wolfe 条件的算法,即为强回溯线性搜索法[6]。在文献[4]中给出了该算法的实现,将该算法分为两个阶段,第一个阶段测试较大的步长以包含区间 $[a_{k-1}, a]$,保证 Wolfe 条件的步长。当满足一下条件之一时,一定包含满足 Wolfe 条件的步长区间:

$$f(x + a^{(k)} d) \geqslant f(x) \tag{11}$$

$$f(x^{(k)} + a^{(k)} d_k) > f(x^{(k)}) + \beta\alpha^{(k)} \nabla_{d^{(k)}} f(x^{(k)}) \tag{12}$$

$$\nabla f(x + a^{(k)} d) \geqslant 0 \tag{13}$$

满足式(12)相当于违反第一个 Wolfe 条件,缩小步长可以确保步长合适。类似地,式(11)和式(13)保证下降步长超过局部最小值,因此它们之间的区域必须包含合适的步长。最后用二分法插值找到满足强 Wolfe 条件的 α。

2　改进的 DFP 算法

2.1　改进的拟牛顿方程

2003 年 Wei 等[7]用 3 种不同的线搜索方法研究了函数:

$$f_k(x) = f(x) - \frac{1}{2}(x - x^{(k)})^{\mathrm{T}} \boldsymbol{A}_k (x - x^{(k)}) \tag{14}$$

式中 $\boldsymbol{A}_k$ 是一个低秩的对称正定矩阵,如果对第 k 次迭代中用 f_k 来计算拟牛顿方程,即 $f(x) := f(x) - \frac{1}{2}(x - x^{(k)})^{\mathrm{T}} \boldsymbol{A}_k (x - x^{(k)})$,通过式(2)有:

$$f(x^{(k+1)} + s) = f(x^{(k+1)}) + \nabla f(x^{(k+1)})^{\mathrm{T}} s + \frac{1}{2} s^T \nabla^2 f(x^{(k+1)}) s - \frac{1}{2} s^{\mathrm{T}} \boldsymbol{A}_k s$$

式中 $s = (x - x^{(k+1)})$,有:

$$g(x^{(k+1)}) - g(x^{(k)}) + \boldsymbol{A}_k (x^{(k+1)} - x^{(k)}) = \boldsymbol{G}(x^{(k)})(x^{(k+1)} - x^{(k)})$$

在此基础上推导出新的拟牛顿方程:

$$\boldsymbol{H}_{k+1} y^* = s_k \tag{15}$$

式中,$y_k^* = y_k + \boldsymbol{A}_k s_k$,$y_k = g(x^{(k+1)}) - g(x^{(k)})$,$s_k = (x^{(k+1)} - x^{(k)})$,在后文用 g_k 表示 $g(x^{(k+1)})$,用 f_k 表示 $f(x^{(k)})$。不难发现当 k 充分大时 $s_k \to 0$,此时新的拟牛顿方程充分接近原始拟牛顿方程。应用这个新的拟牛顿方程,可以推导出一种改进的 DFP 校正公式,即:

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k - \frac{\boldsymbol{H}_k y_k^* (y_k^*)^{\mathrm{T}} \boldsymbol{H}_k}{(y_k^*)^{\mathrm{T}} \boldsymbol{H}_k y_k^*} + \frac{s_k s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k^*} \tag{16}$$

对 $\boldsymbol{A}$ 的选取:

首先假设目标函数 $f(x)$ 足够平滑。现在对目标函数 $f(x)$ 使用泰勒公式,有:

$$f(x) \simeq f_{k+1} + g_{k+1}^{\mathrm{T}}(x - x_{k+1}) + \frac{1}{2}(x - x_{k+1})^{\mathrm{T}} \boldsymbol{G}_{k+1} (x - x_{k+1})$$

式中 $\boldsymbol{G}_{k+1}$ 表示在 x_{k+1} 处的 Hessian 矩阵,因此

$$f_k \approx f_{k+1} - g_{k+1}^T s_k + \frac{1}{2} s_k^T \boldsymbol{G}_{k+1} s_k$$

因此

$$s_k^T \boldsymbol{G}_{k+1} s_k \approx 2(f_k - f_{k+1}) + 2g_{k+1}^T s_k = 2(f_k - f_{k+1}) + (g_{k+1} + g_k)^T s_k + s_k^T y_k$$

又由 $\boldsymbol{H}_{k+1} y_k = s_k$ 有：

$$s_k^T \boldsymbol{H}_{k+1}^{-1} s_k = s_k^T y_k^* = s_k^T y_k + s_k^T \boldsymbol{A}_k s_k$$

上两式相结合表明 $\boldsymbol{A}_k$ 的合理选择应满足以下新的拟牛顿方程：

$$s_k^T \boldsymbol{A}_k s_k = \upsilon_k, (\upsilon_k = 2(f_k - f_{k+1}) + (g_{k+1} + g_k)^T s_k)$$

因此取 $\boldsymbol{A}_k = \dfrac{2(f_k - f_{k+1}) + (g_{k+1} + g_k)^T s_k}{\|s_k\|^2}$。

2.2 一种改进的线性搜索

在数值实验中，由于计算精度原因，算法会存在 $\|g_k\| > \varepsilon$，即并未满足成功条件，但是在给定精度下却没有 α 使得 $f(x+\alpha d)$ 满足强 Wolfe 条件。事实上，在当前精度下也不满足 Armijo 条件。α 在区间 $[0, b]$，$(b>0)$ 上都有 $f(x) = f(x+\alpha d)$，如图 1 所示。出现这种情况时，就无法计算 α，导致算法失败。

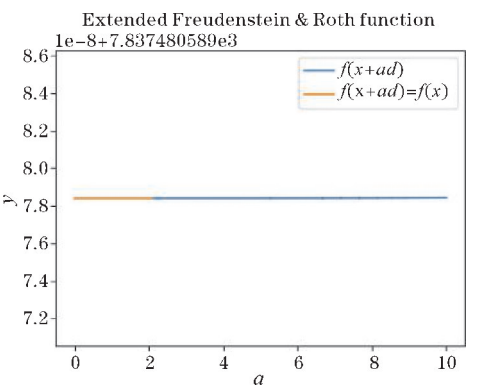

图 1 Extended Freudenstein & Roth function 在迭代中会出现的情况

经实验发现通过调整数据类型，比如 32 位的数据调整为 64 位的数据能改善这种情况，但是在实际应用中并不能无限地调大数据类型。因此，当线搜索失败的时候，采用回溯线搜索，寻找满足条件的 α。

$$f(x+\alpha d) \leqslant f(x) \tag{17}$$

条件(17)是一个比 Armijo 条件更弱的条件，希望当线性搜索失败时，算法能够更积极地去改变 $x^{(k)}$，从而摆脱这种情况。

改进的线性搜索算法：

步骤 1 给出 $\alpha \in \mathbf{R}^*$，$\alpha_{\text{prev}} := 0$，$0 < \beta < \sigma < 1$。

步骤 2 如果 $f(x+\alpha d)$ 满足 Wolfe 条件，则停止。

步骤 3 若 α 满足不等式 $f(x+\alpha d) \geqslant f(x)$，令 $\alpha_{lo} := \alpha_{\text{prev}}$，$\alpha_{hi} := \alpha$ 转步骤 4；若 α 满足不等式 $\nabla f(x+\alpha d) \geqslant 0$ 令 $\alpha_{lo} := \alpha$，$\alpha_{hi} := \alpha_{\text{prev}}$ 转步骤 4；否则令 $\alpha_{\text{prev}} := \alpha$，$\alpha := 2\alpha$ 转步骤 2。

步骤 4 在区间 $[\alpha_{lo}, \alpha_{hi}]$ 中使用二分法寻找满足 Wolfe 条件的 α 并停止；若未能在给定步数内找到 α 则转步骤 5。

步骤 5 令 $\alpha := 1$，$n \in \mathbf{Z}^*$，$i = 0$。

步骤 6 若 α 满足式(17)则搜索成功退出算法；若 $i \geqslant n$ 搜索失败；否则转步骤 7。

步骤 7 令 $\alpha := \dfrac{\alpha}{10}$，$i := i+1$ 转步骤 6。

2.3 改进的 DFP 算法

算法 1：

步骤 1 给出 $x^{(0)} \in R^n$，$\boldsymbol{H}_0 \in R^{n \times n}$，$0 \leqslant \varepsilon$，$k := 0$。

步骤 2 如果 $\|g_k\| \leqslant \varepsilon$，则停止；否则，计算 $d_k := -\boldsymbol{H}_k g_k$。

步骤 3 利用强 Wolfe 准则获取步长 $a^{(k)}$，令 $x^{(k+1)} := x^{(k)} + a^{(k)} d_k$。

步骤 4 计算，用式(16)修正 $\boldsymbol{H}_k$ 得到 $\boldsymbol{H}_{k+1}$。

步骤 5 $k := k+1$，转步骤 2。

算法 2：

步骤 1 给出 $x^{(0)} \in R^n$，$\boldsymbol{H}_0 \in R^{n \times n}$，$0 \leqslant \varepsilon$，$k := 0$。

步骤 2 如果 $\|g_k\| \leqslant \varepsilon$，则停止；否则，计算 $d_k = -\boldsymbol{H}_k g_k$。

步骤 3 利用改进的线性搜索获取步长 $a^{(k)}$，令 $x^{(k+1)} := x^{(k)} + a^{(k)} d_k$。

步骤 4 计算，用式(2.3)修正 $\boldsymbol{H}_k$ 得到 $\boldsymbol{H}_{k+1}$。

步骤 5 $k := k+1$，转步骤 2。

2.4 收敛性分析

一般很难保证梯度型数值方法能够在有限步内得到问题的最优解。人们希望算法能够在每一步的迭代中都有靠近最优点的趋势，最终能在有限步数内迭代到误差所允许范围内的最优点，因此催生出算法收敛性的概念。算法的收敛是指经过多步迭代之后，得出的数值不应该无限增大，而是趋于某个数值。对于无约束最优化问题，一个好的数值方法应该具有良好的收敛性。通过数值实验拟牛顿方法对比最速下降法和牛顿法具有更好的数值效果，但是因为拟牛顿法引入了校正公式，使得拟牛顿法的收敛性证明变得非常复杂。对于一般的非线性函数，人们还不能找到一种拟牛顿算法，使其有全局收敛性。本文对改进算法的收敛性进行分析，证明在一定条件下算法具有全局收敛性。

为证明改进算法的收敛性，需要有如下定义和假设。

定义[18]　设 $\boldsymbol{G}$ 是 $n\times n$ 对称正定矩阵,d_1,d_2 是 n 维非零向量,如果 $d_1^{\mathrm{T}}\boldsymbol{G}d_2=0$ 则称向量 d_1 和 d_2 是 $\boldsymbol{G}$-共轭的。类似的,设 $d_1,d_2,\cdots,d_m$ 是 $\boldsymbol{R}^n$ 中任一组非零向量,如果

$$d_i^{\mathrm{T}}\boldsymbol{G}d_j=0,(i\neq j)$$

则称 $d_1,d_2,\cdots,d_m$ 是 $\boldsymbol{G}$-共轭的。

显然如果 $d_1,d_2,\cdots,d_m$ 是 $\boldsymbol{G}$-共轭的,则它们是线性无关的。如果 $\boldsymbol{G}=I$,则共轭性等价于通常的直交性。

假设:

(a) $f(x)$ 连续可微,x^* 是 $f(x)$ 的极小点;

(b) 有界水平集 $\Omega=\{x\mid f(x)\leqslant f(x_0)\}$ 是凸集,其中 x_0 是给定的初始点;

(c) $g(x)$ 在水平集 Ω 上 Lipschitz 连续,即存在一个常数 $L>0$ 使得下式成立:

$$\|g(x)-g(y)\|\leqslant L\|x-y\|,\forall x,y\in\Omega$$

式中,$g(x)=\nabla f(x)$,$\|\cdot\|$ 是 Euclidean 范数;

(d) $f(x)$ 一致凸,即存在常数 m 和 M 使得

$$m\|z\|^2\leqslant z^{\mathrm{T}}\boldsymbol{G}(x)z\leqslant M\|z\|^2$$

式中 $\forall x\in\Omega$, $z\in R^n$,$\boldsymbol{G}$ 是 $f(x)$ 的 Hessian 矩阵。

(e) $\boldsymbol{G}(x)$ 在 Ω 上 Lipschitz 连续,即存在 $L'>0$ 使得 $\forall x,y\in\Omega$,有:

$$\|\boldsymbol{G}(x)-\boldsymbol{G}(y)\|\leqslant L'\|x-y\|$$

由上面的假设容易得:$\|y_k\|=\|g_{k+1}-g_k\|\leqslant L\|s_k\|$ 且 $\|s_k\|$ 有界 $\lim\limits_{x\to\infty}f(x_k)=f(x^*)$。

定理 1　当且仅当 $s_k^{\mathrm{T}}y_k>0$ 时,式(16)保持正定性。

证明:充分性

选择 $\boldsymbol{H}_0=I$,显然 $z^{\mathrm{T}}\boldsymbol{H}_0z>0$ 对 $\forall z\neq0$ 成立,即 $\boldsymbol{H}_0$ 正定。假设对某个 $k>0$,有 $z^{\mathrm{T}}\boldsymbol{H}_kz>0$ 对 $\forall z\neq0$ 也成立,并记 $\boldsymbol{H}_k=LL^{\mathrm{T}}$ 为 $\boldsymbol{H}_k$ 的 Cholesky 分解。设

$$a=L^{\mathrm{T}}z,b=L^{\mathrm{T}}y_k^*$$

则:

$$\begin{aligned}z^{\mathrm{T}}\boldsymbol{H}_{k+1}z&=z^{\mathrm{T}}\left(H_k-\frac{\boldsymbol{H}_ks_ks_k^{\mathrm{T}}\boldsymbol{H}_k}{s_k^{\mathrm{T}}H_ks_k}\right)z+z^{\mathrm{T}}\frac{s_ks_k^{\mathrm{T}}}{s_k^{\mathrm{T}}y_k^*}z\\&=\left[a^{\mathrm{T}}a-\frac{(a^{\mathrm{T}}b)^2}{b^{\mathrm{T}}b}\right]+\frac{(z^{\mathrm{T}}s_k)^2}{s_k^{\mathrm{T}}y_k^*}\end{aligned}\tag{18}$$

由柯西不等式知:

$$a^{\mathrm{T}}a-\frac{\|a^{\mathrm{T}}b\|^2}{b^{\mathrm{T}}b}\geqslant0\tag{19}$$

又由于题设 $s_k^{\mathrm{T}}y_k>0$,故式(18)中第二项非负,从而

$$z^{\mathrm{T}}\boldsymbol{H}_{k+1}z\geqslant0\tag{20}$$

由于 $z\neq0$,因此在式(19)中当且仅当 a 与 b 平行等式成立,即 z 与 y_k^* 平行,此时有 $z=\beta y_k^*$,$\beta\neq0$,这时

$$\frac{(z^{\mathrm{T}}s_k)^2}{s_k^{\mathrm{T}}y_k^*}=\beta^2s_k^{\mathrm{T}}y_k^*>0$$

即,当 z 与 y_k^* 平行时,式(18)中第二项严格大于 0,于是对任意 $z\neq0$,总有

$$z^{\mathrm{T}}\boldsymbol{H}_{k+1}z>0$$

成立。

必要性

由于 $\boldsymbol{H}_{k+1}$ 正定,即对 $\forall z\neq0$ 有 $z^{\mathrm{T}}\boldsymbol{H}_{k+1}z>0$,并记 $\boldsymbol{H}_k=LL^{\mathrm{T}}$ 为 $\boldsymbol{H}_k$ 的 Cholesky 分解。设

$$a=L^{\mathrm{T}}z,b=L^{\mathrm{T}}y_k^*$$

则:

$$\begin{aligned}z^{\mathrm{T}}\boldsymbol{H}_{k+1}z&=z^{\mathrm{T}}\left(\boldsymbol{H}_k-\frac{\boldsymbol{H}_ks_ks_k^{\mathrm{T}}\boldsymbol{H}_k}{s_k^{\mathrm{T}}\boldsymbol{H}_ks_k}\right)z+z^{\mathrm{T}}\frac{s_ks_k^{\mathrm{T}}}{s_k^{\mathrm{T}}y_k^*}z\\&=\left[a^{\mathrm{T}}a-\frac{(a^{\mathrm{T}}b)^2}{b^{\mathrm{T}}b}\right]+\frac{(z^{\mathrm{T}}s_k)^2}{s_k^{\mathrm{T}}y_k^*}\end{aligned}$$

由柯西不等式知:

$$a^{\mathrm{T}}a-\frac{(a^{\mathrm{T}}b)^2}{b^{\mathrm{T}}b}\geqslant0$$

假设 $s_k^{\mathrm{T}}y_k^*\leqslant0$,此时存在 a 与 b 平行,即 z 与 y_k^* 平行使得 $a^{\mathrm{T}}a-\frac{(a^{\mathrm{T}}b)^2}{b^{\mathrm{T}}b}=0$,这时 $\frac{(z^{\mathrm{T}}s_k)^2}{s_k^{\mathrm{T}}y_k^*}=\beta^2s_k^{\mathrm{T}}y_k^*<0$,这就使得 $z^{\mathrm{T}}\boldsymbol{H}_{k+1}z\leqslant0$,与题设矛盾,所以 $s_k^{\mathrm{T}}y_k^*>0$。

引理 1[8]　对于正定二次函数,共轭方向法至多经过 n 步精确线性搜索终止;且每一 $x^{(i+1)}$ 都是 $f(x)$ 在 x_0 和方向 $d_0,d_1,\cdots,d_i$ 所张成的线性流形 $\{x\mid x=x_0+\sum\limits_{j=0}^{i}\alpha_jd_j,\forall\alpha_j\}$ 中的极小点。

定理 2　如果 f 是二次函数,$\boldsymbol{G}$ 是其正定 Hessian 矩阵,最优步长规则下,改进的 DFP 方法具有遗传性质和方向共轭性质,即对于 $i=0,1,\cdots,m$,有

$$\boldsymbol{H}_{i+1}y_j^*=s_j,j=0,1,\cdots,i(\text{遗传性质})\tag{21}$$

$$s_i^{\mathrm{T}}\boldsymbol{G}s_j=0,j=0,1,\cdots,i-1(\text{方向共轭性})\tag{22}$$

方法在 $m+1\leqslant n$ 步迭代后终止。

证明:显然,当 $i=0$ 时结论成立。现在假定结论对于 i 成立。由于 $g_{i+1}\neq0$,由精确以为搜索和归纳法假设,对于 $j\leqslant i$,有

$$\begin{aligned}g_{i+1}^{\mathrm{T}}s_j&=g_{i+1}^{\mathrm{T}}s_j+\sum_{k=j+1}^{i}(g_{k+1}-g_k)^{\mathrm{T}}s_j\\&=g_{i+1}^{\mathrm{T}}s_j+\sum_{k=j+1}^{i}y_k^{\mathrm{T}}s_j\\&=0+\sum_{k=j+1}^{i}s_j^{\mathrm{T}}\boldsymbol{G}s_j\\&=0\end{aligned}\tag{23}$$

从而利用归纳法假设式(21)和式(23)得到

$$\begin{aligned}s_{i+1}^{\mathrm{T}}\boldsymbol{G}s_j&=-\alpha_{i+1}g_{i+1}^{\mathrm{T}}\boldsymbol{H}_{i+1}y_j\\&=-\alpha_{i+1}g_{i+1}^{\mathrm{T}}s_j\\&=0\end{aligned}\tag{24}$$

这证明了对于 $i+1$,式(22)成立。

设式(21)成立,由公式(16)得

$$\boldsymbol{H}_{i+2}y_{j+1}^{*}=s_{j+1} \tag{25}$$

对于 $j\leqslant i$,由式(24)和式(21),有

$$s_{i+2}^{\mathrm{T}}y_i^{*}=s_{i+1}^{\mathrm{T}}\boldsymbol{G}s_j=0$$

$$y_{i+1}^{*\mathrm{T}}\boldsymbol{H}_{i+1}y_j^{*}=y_{i+1}^{*\mathrm{T}}s_j=s_{i+1}^{\mathrm{T}}\boldsymbol{G}s_j=0$$

故

$$\begin{aligned}\boldsymbol{H}_{i+2}y_j^{*}&=\boldsymbol{H}_{i+1}y_j^{*}+\frac{s_{i+1}s_{i+1}^{\mathrm{T}}y_j^{*}}{s_{i+1}^{\mathrm{T}}y_{i+1}^{*}}-\frac{\boldsymbol{H}_{i+1}y_j^{*}(\boldsymbol{H}_{i+1}y_j^{*})^{\mathrm{T}}}{(y_j^{*})^{\mathrm{T}}\boldsymbol{H}_{i+1}y_j^{*}}\\&=\boldsymbol{H}_{i+1}y_j^{*}\\&=s_j\end{aligned} \tag{26}$$

式(25)和式(26)表明 $\boldsymbol{H}_{i+2}y_j^{*}=s_j$,$j=0,1,\cdots,i+1$。从而式(21)也得证。

由于 s_j 共轭,$i=0,1,\cdots,m$,故方法是共轭方向法。根据共轭方向法基本定理,对于二次函数,共轭方向法至多 n 步终止。

引理 2[1] 若假设(a),(b),(c)成立,由算法 1 产生的点列为$\{x_k\}$,则$\lim\limits_{k\to\infty}\|s_k\|=0$,其中 $s_k=x_{k+1}-x_k$。

引理 3[1] 若假设(b),(c),(d)成立,则下列不等式成立

$$\frac{(y_k^{*})^{\mathrm{T}}s_k}{s_k^{\mathrm{T}}s_k}\geqslant m',\frac{(y_k^{*})^{\mathrm{T}}y_k^{*}}{(y_k^{*})^{\mathrm{T}}s_k}\leqslant M'$$

引理 4[1] 若假设(b),(c)成立,且$\{x_k\}$是由算法 1 产生的点列,则存在正常数 β,γ 对任意的 k 使得下面的结论成立:

$$\|\boldsymbol{H}_k^{-1}s_k\|\leqslant\beta\|s_k\|$$

$$s_k^{T}\boldsymbol{H}_k^{-1}s_k\geqslant\gamma\|s_k\|^2$$

引理 5[1] 设目标函数 $f(x)$ 在 R^n 上满足假设(a),(c)则 Wolfe 步长规则下的下降算法产生的点列$\{x_k\}$满足

$$\sum_{k=1}^{\infty}\|g_k\|^2\cos^2(d_k-g_k)<\infty$$

定理 3 设 x_0 为任意给定初始点,$f(x)$ 满足假设(a),(d),则对任意对称正定矩阵 $\boldsymbol{H}_0$,由算法 1 与算法 2 产生的点列$\{x_k\}$收敛到$f(x)$的极小点 x^{*}。

证明:记 $m_k=\dfrac{(y_k^{*})^{\mathrm{T}}s_k}{s_k^{\mathrm{T}}s_k}$,$M_k=\dfrac{(y_k^{*})^{\mathrm{T}}y_k^{*}}{(y_k^{*})^{\mathrm{T}}s_k}$,由引理 3 可知 $m_k\geqslant m'$,$M_k\leqslant M'$。对式(15)有

$$\mathrm{tr}(\boldsymbol{H}_{k+1})=\mathrm{tr}(\boldsymbol{H}_k)-\frac{\|\boldsymbol{H}_ky_k^{*}\|^2}{(y_k^{*})^{\mathrm{T}}\boldsymbol{H}_ky_k^{*}}+\frac{\|s_k\|^2}{(y_k^{*})^{\mathrm{T}}s_k}$$

$$\det(\boldsymbol{H}_{k+1})=\det(\boldsymbol{H}_k)\frac{(y_k^{*})^{\mathrm{T}}s_k}{(y_k^{*})^{\mathrm{T}}\boldsymbol{H}_ky_k^{*}}$$

令 $\cos\theta_k=\dfrac{(y_k^{*})^{\mathrm{T}}\boldsymbol{H}_ky_k^{*}}{\|y_k^{*}\|\|\boldsymbol{H}_ky_k^{*}\|}$,$q_k=\dfrac{(y_k^{*})^{\mathrm{T}}\boldsymbol{H}_ky_k^{*}}{(y_k^{*})^{\mathrm{T}}y_k^{*}}$,有:

$\dfrac{\|\boldsymbol{H}_ky_k^{*}\|^2}{(y_k^{*})^{\mathrm{T}}\boldsymbol{H}_ky_k^{*}}=\dfrac{q_k}{\cos^2\theta_k}$。

即 $\det(\boldsymbol{H}_{k+1})=\det(\boldsymbol{H}_k)\dfrac{1}{q_kM_k}$。

令 $\psi(\boldsymbol{H}_{k+1})=\mathrm{Tr}(\boldsymbol{H}_{k+1})-\ln(\det(\boldsymbol{H}_{k+1}))$,有:

$$\psi(\boldsymbol{H}_{k+1})=\mathrm{Tr}(\boldsymbol{H}_k)-\frac{q_k}{\cos^2\theta_k}+\frac{1}{m_k}-\ln(\det(\boldsymbol{H}_k))+\ln M_k+\ln q_k$$

$$\psi(\boldsymbol{H}_{k+1})=\psi(\boldsymbol{H}_k)+(\frac{1}{m_k}-\ln\frac{1}{M_k}-1)+(1-\frac{q_k}{\cos^2\theta_k}+\ln\frac{q_k}{\cos^2\theta_k})+\ln\cos^2\theta_k$$

因为函数 $p(t)=1-t+\ln t$ 对任意的 $t>0$ 都是非正的,所以有:

$$0<\psi(\boldsymbol{H}_{k+1})<\psi(\boldsymbol{H}_0)+c(k+1)+\sum_{j=0}^{k}\ln\cos^2\theta_j$$

不失一般性,假设 $c=\dfrac{1}{m'}-\ln\dfrac{1}{M'}-1>0$。假设 $\cos\theta_j\to0$,那么存在 $k_1>0$,使得对 $\forall j>k_1$ 都有 $\ln\cos^2\theta_j<-2c$,所以

$$\psi(\boldsymbol{H}_0)+c(k+1)+\sum_{j=0}^{k_1}\ln\cos^2\theta_j+\sum_{j=k_1+1}^{k}(-2c)$$

$$=\psi(\boldsymbol{H}_0)+\sum_{j=0}^{k_1}\ln\cos^2\theta_j+2ck_1+c-ck$$

对充分大的 k,上式右端为负,矛盾。从而存在一个子序列$\{j_k\}$使得 $\cos\theta_j\geqslant\varepsilon>0$。而由引理 5 可知$\liminf\|\nabla f_k\|\to0$. 又目标函数是凸函数,稳定点就是唯一最小值点,从而点列$\{j_k\}$收敛到目标函数的唯一极小值点。

3 数值实验

将对本文中提出的改进的 DFP 算法进行数值实验,并与经典的 DFP 算法进行数值比较,算法用 Python 编程实现。数值实验在 PC 机上 Windows11 系统中进行,CPU 为 Inteli7-8750H,主频2.20 GHz,内存16.0 G,程序运行环境是Python3.9。

算法中参数的选择如下:

$$\varepsilon=10^{-6},\delta_1=10^{-4},\delta_2=0.9$$

式中 ε 为终止条件,即$\|g\|<\varepsilon$ 时,算法终止。δ_1,δ_2 为强 Wolfe 准则即式(8)与(10)中的参数。

本文采用的测试算例主要来自文献[9],所有的偏导计算均采用自动微分实现。经过测试发现,计算 1 次微分的时间是计算一次函数的 4 倍左右,因此用

公式 N_total=NF+mNG 来评估总的函数计算,其中 NF 是函数计算次数,NG 是微分计算次数并且设定 $m=4$。

最终,计算结果由表 1 来表示,其中:functions 表示测试函数名称;dims 表示测试函数的维数;time 表示求解函数需要的时间;N_total 表示函数评估计算次数。

采用 ED Dolan 和 JJ Moré 提供的方法[9]来比较原始 DFP 算法,算法 1-DFP1,算法 2-DFP2 这 3 种算法。即设定算法集 S 和测试集 P,算法数量和问题数量分别为 n_S 和 n_P,并设定 $t_{p,s}$ 为用算法 $s\in S$ 去求解问题 $p\in P$ 所需要的时间(或者函数计算次数)。设定性能比 $r_{p,s}$ 为

$$r_{p,s}=\frac{t_{p,s}}{\min\{t_{p,s}:s\in S\}}$$

当且仅当 s 不能求解 p 时,设定 $r_{p,s}=r_M$ 其中,对 $\forall s\in S, p\in P$ 有 $r_M\geqslant r_{p,s}$,r_M 的取值不会影响评估结果。最后计算

$$\rho_s(\tau)=\frac{1}{n_p}\text{size}\{p\in P:\log_2(r_{p,s})<\tau\}$$

式中 $\rho_s(\tau)$ 为性能比的分布函数,$\rho_s(\tau)\leqslant 1$,$\rho_s(\tau)$ 越大,代表性能越好。

表 1　运算结果表

function	dim	DFP		DFP1		DFP2	
		time	N_total	time	N_total	time	N_total
ARWHEAD	4	0.077787	150	0.100163	165	0.101172	165
	20	0.101155	169	0.093776	149	0.094652	149
	80	0.094821	173	0.135679	191	0.135169	191
	160	–	–	0.13899	192	0.13907	192
	320	0.159317	213	0.167065	193	0.185158	193
Broyden Tridiagonal	4	0.674463	1044	0.25288	337	0.255585	337
	20	0.765022	1141	0.51423	688	0.535645	688
	80	1.404885	1951	2.505933	3059	2.504523	3059
	160	2.095384	2817	3.4067	4013	3.389313	4013
	320	5.194672	5649	7.137379	6823	7.235965	6823
CRAGGLVY	4	4.368444	7607	1.01631	1435	0.997689	1435
	20	5.485934	9219	8.861196	12237	8.700827	12237
	80	–	–	–	–	12.58749	16933
	160	–	–	–	–	–	–
	320	–	–	–	–	–	–
Diagonal 5	4	0.024984	59	0.022999	46	0.023655	46
	20	0.026994	59	0.037933	65	0.03677	65
	80	0.034983	59	0.036984	65	0.036568	65
	160	0.025843	59	0.037686	65	0.038	65
	320	0.038023	59	0.050877	65	0.052586	65
EG2	4	0.316693	567	0.476413	692	0.50461	692
	20	0.16861	289	0.192849	291	0.19619	291
	80	0.281084	453	0.381325	495	0.361174	495
	160	0.341564	534	0.298115	402	0.293615	402
	320	0.42328	519	0.319351	332	0.368304	332
Extended BD1	4	0.233532	448	0.102869	169	0.102118	169
	20	0.320178	567	0.10445	169	0.100595	169
	80	0.336078	575	0.114843	169	0.118931	169
	160	0.317495	549	0.114754	169	0.117523	169
	320	0.509652	636	0.180079	209	0.182268	209
Extended Beale	4	0.186984	340	0.208021	294	0.19704	294
	20	0.291207	489	0.257951	361	0.248093	361
	80	0.317698	488	0.27301	359	0.270423	359
	160	0.309479	477	0.282733	366	0.274424	366
	320	0.47156	579	0.422916	457	0.420602	457
Extended Freudenstein & Roth	4	0.133239	245	0.14576	236	0.136348	236
	20	0.212282	385	–	–	–	–
	80	–	–	–	–	–	–
	160	–	–	–	–	0.59628	612
	320	0.462056	417	0.430223	314	0.40322	314

续表 1

function	dim	DFP		DFP1		DFP2	
		time	N_total	time	N_total	time	N_total
Extended PSC1	4	0.168732	307	0.164071	262	0.159185	262
	20	0.193452	342	0.167075	262	0.164387	262
	80	0.179815	333	0.201756	303	0.202057	303
	160	0.233312	362	0.223463	303	0.214358	303
	320	0.293087	343	0.301459	303	0.290498	303
Extended Penalty	4	0.164275	315	0.282636	435	0.272987	435
	20	8.002553	16136	1.940862	3044	1.915161	3044
	80	–	–	–	–	–	–
	160	–	–	–	–	–	–
	320	–	–	–	–	–	–
Extended Powell	4	75.89235	150652	14.34405	22950	14.38247	22950
	20	–	–	72.19446	116771	72.3046	116771
	80	–	–	–	–	–	–
	160	–	–	–	–	–	–
	320	–	–	–	–	–	–
Extended TET	4	0.069064	148	0.092684	164	0.094523	164
	20	0.078301	165	0.099561	164	0.101298	164
	80	0.081705	165	0.102333	164	0.099455	164
	160	0.088039	165	0.110511	164	0.117267	164
	320	0.147535	185	0.143621	164	0.141521	164
Extended Tridiagonal 2	4	0.062598	111	0.059983	104	0.060522	104
	20	0.177562	336	0.255216	399	0.251808	399
	80	0.401783	728	0.523131	793	0.701694	793
	160	0.429307	675	0.608468	831	0.671608	831
	320	0.624003	712	0.744273	732	0.713897	732
Extended Wood	4	13.39381	24133	12.39518	18421	13.42438	18421
	20	24.28411	42377	23.56346	32716	22.67774	32716
	80	28.29388	45684	25.43207	34147	25.29694	34147
	160	30.85096	47365	35.27938	45511	35.50232	45511
	320	44.37062	52740	38.38378	39246	38.39842	39246
Extended quadratic penalty QP2	4	0.196234	360	0.319002	476	0.305198	476
	20	0.248641	470	0.402586	624	0.402072	624
	80	0.29014	489	0.510356	710	0.491007	710
	160	8.796805	14682	0.99746	1349	1.014403	1349
	320	0.847718	1060	32.069	32734	31.12291	32734
FLETCHCR	4	0.185206	298	0.157086	231	0.164234	231
	20	0.953809	1797	1.236707	1821	1.411616	1821
	80	54.21851	97668	6.443769	8474	6.159285	8474
	160	24.52221	40759	40.22495	53367	40.39655	53367
	320	–	–	66.15852	71733	65.08671	71733
Generalized PSC1	4	38.14198	74006	0.812593	1315	1.045305	1315
	20	–	–	74.22465	120417	73.83149	120417
	80	–	–	–	–	–	–
	160	–	–	–	–	–	–
	320	–	–	–	–	–	–
LIARWHD	4	0.3222	563	0.359478	539	0.371823	539
	20	0.729188	1354	0.25581	373	0.255392	373
	80	0.618892	1057	0.500215	724	0.504315	724
	160	1.030214	1660	0.596758	743	0.567165	743
	320	1.043422	1293	0.721431	753	0.711189	753
Partial Perturbed Quadratic	4	0.117198	183	0.15379	203	0.155177	203
	20	0.642127	548	0.760407	602	0.739773	602
	80	5.642218	1942	6.848748	2126	6.535949	2126
	160	19.45692	3498	20.19344	3818	20.12043	3818
	320	54.58624	5382	55.07685	5868	54.41143	5868

通过图 2 可以明显看出,改进后的两种算法的函数计算次数要明显少于原始 DFP 尽管如此,从图 3 可以看出,改进过后的算法的计算时间在 $0<\tau<2^{-4}$ 时要明显高于原始 DFP 算法并在 $2^{-4}<\tau<2^{-1}$ 时,改进的两种算法与原始的算法在时间上区别不大,总体上要略慢于原始 DFP 算法,但是当 $2^{-1}<\tau$ 时,改进的算法就要明显好于原始的 DFP 算法。通过图 2 和图 3 可以看到改进后算法 1 能比原始的算法多解决 6% 左右的问题,而改进后的算法 2 能多解决 8% 左右的问题。可以认为,通过增加矩阵 $\boldsymbol{A}$ 能够使算法能有更快的迭代速度以及提高算法的收敛性,改进的线性搜索也能够提升算法的收敛性。但是由于 $\boldsymbol{A}$ 需要额外的计算,对于本来计算量很小的算例,提升还不明显。但是对于计算量大的算例,改进的算法就能大大提高收敛速度。

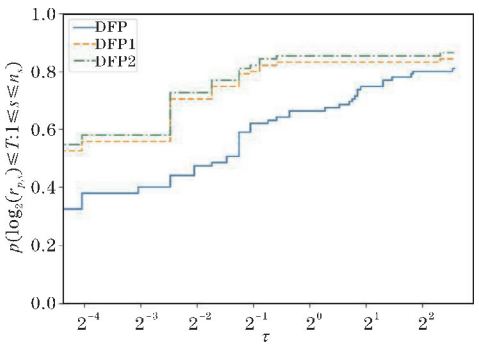

图 2　用函数计算次数来进行评估

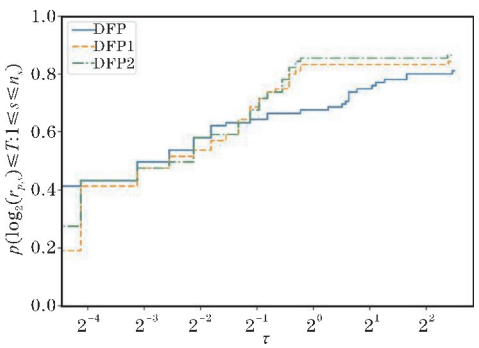

图 3　用 CPU 运行时间来进行评估

4　结束语

根据Wei等[2]提出的拟牛顿方程建立新的拟牛顿公式,提出新的 DFP 校正公式,引入低秩的对称正定矩阵 $\boldsymbol{A}_k$ 使得 $y_k^* = y_k + \boldsymbol{A}_k s_k$。通过研究线性搜索失败的情况,提出一种改进的线性搜索算法。在新的 DFP 校正公式与改进的线性搜索下,提出两种算法。之后本文证明了改进后算法的全局收敛性,并使用算例进行数值实验。数值实验的结果表明,改进的算法 1 要优于原始算法,并且改进的算法 2 相对于算法 1 有更好的收敛性。

参考文献:

[1] 王宜举,修乃华. 非线性最优化理论与方法[M]. 北京:科学出版社,2016.

[2] Wei Z, Li G, Qi L. New quasi-Newton methods for unconstrained optimization problems[J]. Applied Mathematics and Computation, 2006, 175(2): 1156-1188.

[3] Andrei N. An unconstrained optimization test functions collection[J]. Adv. Model. Optim, 2008, 10(1): 147-161.

[4] Kochenderfer M J, Wheeler T A. Algorithms for optimization[M]. Mit Press, 2019.

[5] Armijo L. Minimization of functions having Lipschitz continuous first partial derivatives[J]. Pacific Journal of mathematics, 1966, 16(1): 1-3.

[6] Jorge N, Stephen J W. Numerical optimization[M]. New York, NY: Spring New York, 1999.

[7] Wei Z, Qi L, Chen X. An SQP-type method and its application in stochastic programs[J]. Journal of Optimization Theory and Applications, 2003, 116(1): 205-228.

[8] 袁亚湘,孙文瑜. 最优化理论与方法[M]. 北京:科学出版社,1997.

[9] Dolan E D, Moré J J. Benchmarking optimization software with performance profiles[J]. Mathematical programming, 2002, 91(2): 201-213.

[10] Wu Q, Wei Z. Some new step-size rules for optimization problems[J]. Journal of Shanghai University (English Edition), 2007, 11(2): 135-141.

An Improved Quasi-Newtonian Algorithm

LUO Wenjun[1], WU Zezhong[1], HE Shengyu[2]

(1. College of Applied Mathematics, Chengdu University of Information Technology, Chengdu 610225, China; 2. Sichuan Institute of Socialism, Chengdu 610037, China;)

Abstract: In this paper, based on the new quasi-Newtonian equation, a new DFP correction formula is derived, and a new class of DFP algorithm is proposed under the strong Wolfe condition. Then, we propose an improved strong Wolfe linear search method to solve the problem of linesearch failure due to accuracy, and prove that the improved algorithm has global convergence under certain assumptions. Finally, an example is introduced to compare the performance of the DFP algorithm before and after the improvement. The results show that the improved algorithm is moreeffective and has better convergence.

Keywords: DFP; conjugate gradient; quasi-Newtonian method; unconstrained optimization; linear search