共轭梯度法

我们关注共轭梯度法主要有两个原因

1. 它们是求解大规模矩阵方程最有效的方法之一
2. 它们可以经过一些调整去处理非线性的最优化问题

本章我们将讨论共轭梯度法在处理*线性*和*非线性*问题上的优越性质。

首先，线性共轭梯度法是高斯消元法的变种，它适用于处理大规模矩阵方程问题，它的性能取决于系数矩阵的特征值的分布，因此我们还可以通过一些预条件显著改善这种方法的收敛性。预条件在共轭梯度法的实际应用中非常重要。

基于非线性共轭梯度法的优化算法有个很重要的特征就是他们不要求储存矩阵，且显著快于最速下降法。

线性共轭梯度法

本节我们首先讨论如何用共轭梯度法解决线性问题并研究其收敛性。
它是一种用于求解下述方程(5.1)的迭代方法：

$$Ax = b$$

其中A 是$n \times n$的对称正定阵，而求解方程(5.1)又等价于求解极小化问题(5.2)

$$\min \phi(x) \overset{def}{=} \frac{1}{2}x^T Ax - b^T x$$

这个定价性告诉我们共轭梯度法即是一种求解线性系统的方法，又是一项极小化凸二次函数的技术。
此外，我们注意到，(5.2)的梯度又恰好是线性系统的误差，即(5.3)

$$\nabla \phi(x) = Ax - b = r(x)$$

$x = x_k$时，我们记(5.4)

$$r_k = Ax_k - b$$

共轭方向法

共轭梯度法有一个非常重要的属性就是它的**共轭性**：

> 集合$\{p_1, p_2, \ldots, p_l\}$关于对称正定阵$A$共轭 等价于(5.5)
>
> $$p_i^T Ap_j = 0, \quad \forall i \neq j$$

显然这样的集合是线性独立的。

而共轭性的重要意义就在于，我们可以通过沿着集合内的每一个方向做极小化，从而**在n步内极小化函数ϕ**。
为了验证这个说法，我们考虑*共轭方向法*。给一个初始点x_0和一组共轭方向$\{p_0, p_1, \ldots, p_{n-1}\}$，我们取(5.6)

$$x_{k+1} = x_k + \alpha_k p_k$$

其中α_k 是子问题$\phi(x_k + \alpha p_k)$的极小子，则（5.7）

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T Ap_k}$$

于是我们有如下定理

定理5.1

对于任意的初值x_0，我们用(5.6)(5.7)生成序列$\{x_k\}$在n步内收敛到(5.1)的解x^*处

思路：
由于$\{p_k\}$线性无关，因此它能张满全空间，于是$x^* - x_0 = \sum \sigma_k p_k$，最后证明$\alpha_k = \sigma_k$

显然，当A是对角阵的时候，我们的方法其实是沿着坐标轴去做最优化且显然能达到最优解，而当A非对角的时候，我们可以借助其正定对称的性质，引入共轭方向将其化为对角阵$S^T AS$，问题又变得直观，而共轭方向法也有了直观的解释。
同时这样的解释也给了我们一些新的发现：当Hessian矩阵是对角阵的时候，每次迭代只在某个坐标轴方向上进行，同时每次迭代后，变得到了这个分量的最优解。也就是说，k步迭代后，这个二次型在$e_1, e_2, \ldots, e_k$张成的*子空间*上取到了极小值。
下面的定理将给出一个更普适的结果，首先，我们由记(5.4)和(5.6)可推出(5.10)

$$r_{k+1} = r_k + \alpha_k A p_k$$

定理5.2 张成子空间极小化

任取$x_0 \in R^n$，并由(5.6)(5.7)生成$\{x_k\}$，则有(5.11)

$$r_k^T p_i = 0, \quad \forall i = 0, 1, \ldots, k-1$$

且x_k是下述问题的极小子，即(5.12)

$$x_k = \arg\min \phi(x) \quad \{x | x = x_0 + \mathrm{span}\{p_0, p_1, \ldots, p_{k-1}\}\}$$

思路：
首先分析(5.12)是凸优化问题有唯一极小子x^*，且x^*满足$r\left(x^*\right)^T p_k = 0$,再用归纳法说明(5.11)成立，则定理证明完成。

其中，r_k**垂直于之前所有的方向**的性质将在后续分析中大量使用。
至此，我们的讨论都是很广泛的，因为任意一组具有共轭性质的方向都满足我们的要求。比如特征向量组就是一组很好的共轭方向。然而在大规模问题中，求解特征向量组的运算代价是很大的。有一个解决方法是，我们类似施密特正交化方法去构造共轭方向组，然而，这种方法的代价也很大，因为要储存整个方向组会占用大量的空间。

共轭梯度法的基本性质

共轭梯度法是一种非常特殊的共轭方向法，因为它在计算p_k的时候只需要使用p_{k-1}的信息。这一特征保证了它的计算量和空间占用代价都不会太大。

在共轭梯度法里，每一个迭代方向都是负残差(5.3)r_k也即当前的最速下降方向与上一个方向的线性组合，具体形式为(5.13)

$$p_k = -r_k + \beta_k p_{k-1}$$

其中β_k可以由p_{k-1}与p_k共轭这一条件推出，由(5.13)及共轭性条件我们可以知道

$$\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}$$

初始方向p_0我们设为x_0处的最速下降方向。综上，共轭梯度法可表示为：

Algorithm 5.1 共轭梯度法的初级版本

```
Given x(0)
Set r(0) <- Ax(0)-b, p(0) <- -r(0), k <- 0;
While r(k) != 0,        (5.14)
    alpha(k) <-  -r(k)'p(k)/( p(k)'Ap(k) )
    x(k+1) <- x(k) + alpha(k)p(k)
    r(k+1) <- Ax(k+1) - b
    beta(k+1) <- r(k+1)'Ap(k)/( p(k)'Ap(k) )
    p(k+1) <- -r(k+1) + beta(k+1)p(k)
    k <- k+1
end
```

这个版本在研究共轭梯度法的核心性质的时候是很有用的，但是我们会在后面提供一个效率更高的版本。
首先我们证明$p_0, p_1, \ldots, p_{n-1}$的确是共轭的，然后利用定理5.1我们就可以说明算法至少是n步内收敛的。
下面的定理将说明这个结果，同时还会给出两个其他的重要结果:

1. r_i相互垂直
2. p_k和r_k被包含在关于r_0且自由度为k的Krylov子空间内：(5.15)

$$\mathcal{K}(r_0; k) \stackrel{def}{=} \text{span}\{r_0, Ar_0, \ldots, A^k r_0\}$$

定理5.3

假设共轭梯度法的第k个迭代点仍不是x^*，我们有如下结果：

$$r_k^T r_i = 0, \quad \text{for } i = 0, 1, \ldots, k-1 \qquad (5.16)$$

$$\text{span}\{r_0, r_1, \ldots, r_k\} = \text{span}\{r_0, Ar_0, \ldots, A^k r_0\} \qquad (5.17)$$

$$\text{span}\{p_0, p_1, \ldots, p_k\} = \text{span}\{r_0, Ar_0, \ldots, A^k r_0\} \qquad (5.18)$$

$$p_k^T A p_i = 0, \quad \text{for } i = 0, 1, \ldots, k-1 \qquad (5.19)$$

本定理可由归纳法证明。
然而值得注意的一点是，在这个定理的证明是基于第一个迭代方向p_0是最速下降方向$-r_0$完成的。实际上，选取任何其他的p_0，我们都不能得到这个结果。
由于梯度r_k是相互垂直的，所以共轭梯度法这个名字其实有点不太恰当，事实上在共轭梯度法里，关于A共轭的不是梯度，而是搜索方向。

共轭梯度法的实践算法

利用定理5.2、5.3的结果，我们可以得到运算量更小的共轭梯度算法。

1. 我们利用(5.14e)和(5.11)去替换(5.14a)

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$$

2. 由(5.10)我们有$\alpha_k A p_k = r_{k+1} - r_k$，所以根据(5.14e)和(5.11)，我们可以简化β_k如下

$$\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

利用这两个结果和(5.10)，我们得到了共轭梯度法的标准形式：

Algorithm 5.2 CG

```
Given x(0)
Set r(0) <- Ax(0) - b, p(0) <- -r(0), k <- 0;
while r(k) != 0   (5.24)
    alpha(k) <- (r(k)'r(k))/(p(k)'Ap(k))
    x(k+1) <- x(k) + alpha(k)p(k)
    r(k+1) <- r(k) + alpha(k)p(k)
    beta(k+1) <- (r(k+1)'r(k+1))/(r(k)'r(k))
    p(k+1) <- -r(k+1) + beta(k+1)p(k)
    k <- k+1
end
```

上述算法的核心运算量发生在矩阵向量乘法和向量内积中。
共轭梯度法仅在问题规模大的时候推荐使用，否则高斯消元法或者其他矩阵分解的方法效果更好，因为它们对舍入误差不那么敏感。而对于大规模问题，共轭梯度法有一个优势就是，因为它不改变原矩阵所以不用开辟空间储存矩阵信息，另外，有时共轭梯度法的求解速度很快。

收敛速率

我们看到共轭梯度法能够在至多n次迭代中达到最优解。然而，当A的特征值分布满足一些条件的时候，算法将会在显著小于n次迭代后达到最优解。为了解释这个性质，我们首先换个角度看看定理5.2中提到的子空间最小性，并且用它说明算法5.2在某种意义下是最优的。

由(5.24b)和(5.18)可得(5.25)

$$\begin{aligned} x_{k+1} &= x_0 + \alpha_0 p_0 + \cdots + \alpha_k p_k \\ &= x_0 + \gamma_0 r_0 + \gamma_1 A r_0 + \cdots + \gamma_k A^k r_0 \end{aligned}$$

我们引入$P_k^*(A) = \gamma_0 I + \gamma_1 A + \cdots + \gamma_k A^k$，于是(5.25)可表示为(5.26):

$$x_{k+1} = x_0 + P_k^*(A)r_0$$

我们下面说明：对于那些前k步被限制在(5.15)定义的Krylov子空间内的方法，算法5.2的最小化效果是最好的，这里我们用的范数是$\|\cdot\|_A$，定义如下：(5.27)

$$\|z\|_A^2 = z^T A z$$

这个范数在(5.2)定义的最小化问题中有如下性质：对于ϕ的极小子x^*，我们显然有(5.28)

$$\frac{1}{2}\|x - x^*\|_A^2 = \frac{1}{2}(x - x^*)^T A(x - x^*) = \phi(x) - \phi(x^*)$$

定理5.2说明x_{k+1}在空间$x_0 + \text{span}\{p_0, p_1, \ldots, p_k\}$上极小化了$\phi$，因此也极小化了$\|x - x^*\|_A^2$，利用(5.26)我们有：(5.29)

$$\min_{P_k} \|x_0 + P_k(A)r_0 - x^*\|_A$$

我们之后中会反复研究这个问题的最优性条件。
由于

$$r_0 = Ax_0 - b = Ax_0 - Ax^* = A(x_0 - x^*)$$

我们有(5.30)

$$x_{k+1} - x^* = x_0 + P_k^*(A)r_0 - x^* = [I + P_k^*(A)A](x_0 - x^*)$$

记A的特征值为$0 \le \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$，其对应的特征向量为$v_1, v_2, \ldots, v_n$，于是有

$$A = \sum_{i=1}^n \lambda_i v_i v_i^T$$

由于特征向量可以张成全空间，我们有

$$x - x^* = \sum_{i=1}^n \xi_i v_i$$

不难知，A的任意特征值也是$P_k(A)$的特征值，更具体地，我们有

$$P_k(A)v_i = P_k(\lambda_i)v_i, \quad i = 1, 2, \ldots, n$$

将(5.31)代入(5.30)，我们有

$$x_{k+1} - x^* = \sum_{i=1}^n [1 + \lambda_i P_k^*(\lambda_i)]\xi_i v_i$$

利用公式$\|z\|_A^2 = z^T A z = \sum_{i=1}^n \lambda_i (v_i^T z)^2$，有

$$\|x_{k+1}-x^*\|_A^2=\sum_{i=1}^n\lambda_i[1+\lambda_iP_k^*(\lambda_i)]^2\xi_i^2$$

因为由共轭梯度法产生的P_k^*在这个范数意义下是最优的，我们有

$$\|x_{k+1}-x^*\|_A^2=\min_{P_k}\sum_{i=1}^n\lambda_i[1+\lambda_iP_k(\lambda_i)]^2\xi_i^2$$

于是有(5.33)

$$\begin{aligned}\|x_{k+1}-x^*\|_A^2&\le\min_{P_k}\max_{1\le i\le n}[1+\lambda_iP_k^*(\lambda_i)]^2\left(\sum_{j=1}^n\lambda_i\xi_i^2\right)\\&=\min_{P_k}\max_{1\le i\le n}[1+\lambda_iP_k^*(\lambda_i)]^2\|x_0-x^*\|_A^2\end{aligned}$$

这里用到了$\|x_0-x^*\|_A^2=\sum_{j=1}^n\lambda_j\xi_j^2$

(5.33)说明我们可以用下面的式子去刻画共轭梯度法的收敛速率(5.34)

$$\min_{P_k}\max_{1\le i\le n}[1+\lambda_iP_k(\lambda_i)]^2$$

也就是说我们需要找到一个使上式尽可能小的P_k。、
在实际应用中，我们能显式地找到这个多项式，并且得到一些有趣的结论。其中一个结论如下：

定理5.4

如果A只有r个不同的特征值，那么共轭梯度法最多r次就能终止。

证明：

假设$\lambda_1,\lambda_2,\dots,\lambda_n$里只有$r$个不同的值$\tau_1<\tau_2<\cdots<\tau_r$，则我们定义如下多项式$Q_r(\lambda)$:

$$Q_r(\lambda)=\frac{(-1)^r}{\tau_1\tau_2\cdots\tau_r}(\lambda-\tau_1)(\lambda-\tau_2)\cdots(\lambda-\tau_r)$$

注意到$Q_r(\lambda_i)=0$，且$Q_r(0)=1$，于是$Q_r(\lambda)-1$是一个在原点有根的r次多项式，所以使用多项式除法，我们定义$\bar{P}_{r-1}$为

$$\bar{P}_{r-1}(\lambda)=(Q_r(\lambda)-1)/\lambda$$

则$\bar{P}_{r-1}$是一个$r-1$阶多项式。令(5.34)中的$k=r-1$，我们有

$$0\le\min_{P_{r-1}}\max_{1\le i\le n}[1+\lambda_iP_{r-1}(\lambda_i)]^2\le\max_{1\le i\le n}[1+\lambda_i\bar{P}_{r-1}(\lambda_i)]^2=\max_{1\le i\le n}Q_r^2(\lambda_i)=0$$

于是，当$k=r-1$时，(5.34)中的常数为0，代入(5.33)得，$\|x_r-x^*\|_A^2=0$，从而有$x_r=x^*$

定理5.5

如果A的特征值$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$，我们有

$$\|x_{k+1} - x^*\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1} \right)^2 \|x_0 - x^*\|_A^2$$

这个定理可以借助(5.33)得到

下面我们说明如何利用定理5.5 去预测共轭梯度法在某个具体问题上的效果。假设矩阵A有m个大特征值，且其余$n-m$个特征值均在1附近，若记$\epsilon = \lambda_{n-m} - \lambda_1$由5.5我们知道，在$m+1$次共轭梯度法后，我们有

$$\|x_{m+1} - x^*\|_A^2 \approx \epsilon \|x_0 - x^*\|_A$$

即，共轭梯度法在至少$m+1$次迭代后才能得到一个比较好的估计。

然而注意到，定理5.5只是给出了一个收敛效果的上界，然而其实际收敛速率可能会更快。但是，对于特征值分布比较随机的问题，共轭梯度法的收敛效果比较慢且均匀。

如果特征值的分布是r个簇，那么共轭梯度法将会在大约r步内给出一个近似解。

另一个更粗糙的估计是基于矩阵的欧几里得条件数去进行的

$$\kappa(A) = \|A\|_2 \|A^{-1}\|_2 = \lambda_n / \lambda_1$$

于是我们有

$$\|x_k - x^*\|_A \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|x_0 - x^*\|_A$$

这个估计中的上界给得很宽，但是在那种只知道特征值最值的问题里还是挺有用的。同时，我们可以将它与(3.29)给出的最速下降法的上界进行对比。

预条件

我们可以通过矩阵变换改善特征值的分布从而提高共轭梯度法的效率。这种预条件的手段是通过一个非奇异阵C将x变为$\hat{x}$：(5.37)

$$\hat{x} = Cx$$

于是(5.2)变为

$$\hat{\phi}(\hat{x}) = \frac{1}{2} \hat{x}^T (C^{-T} A C^{-1}) \hat{x} - (C^{-1} b)^T x$$

如果我们使用算法5.2去极小化$\hat{\phi}$，或者去求解等价的线性系统

$$(C^{-T} A C^{-1}) \hat{x} = C^{-1} b$$

那么算法的收敛速率将由$C^{-T}AC^{-1}$ 而非A的特征值决定。因此，我们希望找到使$C^{-T}AC^{-1}$ 的特征值更符合我们要求的C。我们可以去找那些能让$C^{-T}AC^{-1}$ 的条件数远小于A的条件数的C，也可以去寻找让$C^{-T}AC^{-1}$ 的特征值聚集的C。

实际上我们不必显示地调用(5.37)，我们可以直接对问题(5.38)关于$\hat{x}$使用算法5.2，然后在用逆变换得到x。这便是我们在算法5.3里的做法。在算法5.3里，我们并不显式使用C，而是用一个构造出来的对称正定矩阵$M = C^TC$。

Algorithm 5.3 (Preconditioned CG)

```
Given x(0), preconditioner M
Set r(0) <- Ax(0)-b;
Solve My(0) <- r(0) for y(0);
Set p(0) <- -y(0), k <- 0;
While r(k) != 0 (5.39)
    alpha(k) <- (r(k)'y(k))/(p(k)'Ap(k))
    x(k+1) <- x(k) + alpha(k)p(k)
    r(k+1) <- r(k) + alpha(k)Ap(k)
    Solve My(k+1) = r(k+1)
    beta(k+1) <- (r(k+1)'y(k+1))/(r(k)'y(k))
    p(k+1) <- -y(k+1) + beta(k+1)p(k)
    k <- k+1
end
```

在上述算法中令$M = I$，则得到了标准共轭梯度法5.2。算法5.2的性质用一种有趣的方式推广到了这个算法里。特别的，(5.16)中的正交性变成了(5.40)

$$r_i^T M^{-1} r_j = 0, \quad \text{for all } i \neq j$$

在计算量上考虑，5.3和5.2的主要区别在于求解$My = r$

预条件的实践算法

首先，没有任何一个预条件策略对所有问题都是最好的：我们有很多需要考虑的因素——M的有效性、计算和储存的方便性、求解线性方程的简便性

而对于特定的一些矩阵，是存在一些好的预条件算法的，尤其是那些从PDE问题中导出的矩阵。通常，预条件子会用如下方式定义:$My = r$，这是原线性系统的一个简化版本。在PDE问题里，$My = r$通常是原问题的一个更粗糙的离散形式。同样地，在许多其他的优化和数值分析问题中，对原问题结构和起源的知识往往是设计一个高效算法的关键。
也有人提出了一些通用的预条件子，但是这些预条件子的效果往往随问题的改变而改变。这类方法中最重要的策略包括超松弛迭代(SSOR)，不完全Cholesky，和带状预处理法。其中，不完全Cholesky可能通常是最有效的方法。它的基本思想非常简单：我们还是用Cholesky分解的流程，但是我们不去计算精确的Cholesky因子L s.t. $A = LL^T$，而去计算一个比L稀疏的近似因子$\tilde{L}$(通常，我们要求$\tilde{L}$是不比原矩阵的下三角稠密)。于是我们有$A \approx \tilde{L}\tilde{L}^T$，并且取$C = \tilde{L}^T$，我们有$M = \tilde{L}\tilde{L}^T$，且

$$C^{-T}AC^{-1} = \tilde{L}^{-1}A\tilde{L}^{-T} = I$$

于是$C^{-T}AC^{-1}$是比较好的，我们不去显式地计算M，而是去储存$\tilde{L}$并用它去求解线性系统$My = r$。由于$\tilde{L}$的洗属性与A相似，因此计算$My = r$的代价近似于计算矩阵-向量乘法Ap。
然而这种方法也有一些缺陷。其一是其产生的矩阵可能并不充分正定，在这种情况下，我们需要增大对角元素的值去保证$\tilde{L}$能够被找到。我们要求$\tilde{L}$具有稀疏性可能会导致数值求解上不稳定，这一问题可以通过适当增加$\tilde{L}$的元素改善，但是这样的做法也会导致算法在迭代过程中运算量的增加。

非线性共轭梯度法

我们已经提到了共轭梯度法，即算法5.2，可以看做是一个极小化由(5.2)定义的凸二次函数ϕ的方法。我们很自然地联想到能否适当改变这个方法去极小化一般的凸函数，甚至是更一般的非线性函数f。实际上，共轭梯度法的非线性变种已经得到较好地研究并在实际应用中非常有效。

Flecher-Reeves法

Flecher和Reeves 对算法5.2做了两个小改变将算法推广到了非线性函数上：

1. 对于求解步长α_k的(5.24a)，我们利用一个线性搜索法实现；
2. 对于残差r，本来是ϕ的梯度，这里替换为非线性函数f的梯度。

Algorithm 5.4 (FR)

```
Given x(0);
Evaluate f(0) = f(x(0)), df(0) = df(x(0))
Set p(0) <- -df(0), k <- 0;
while df(k) != 0
    Compute alpha(k) and Set x(k+1) = x(k) + alpha(k)*p(k)
    Evaluate df(k+1)
    (5.41):
        beta_FR(k+1) <- (df(k+1)'df(k+1))/(df(k)'df(k))
        p(k+1) <- -df(k+1) + beta_FR(k+1)p(k)
        k <- k+1
end
```

如果f是强凸二次函数且α_k是子问题的精确最小子，那么这个算法退化为线性共轭梯度法。算法5.4是适用于大规模非线性问题的，因为每次迭代只用到了函数值和梯度值，并不要求进行矩阵运算，只需要记录少量的向量信息。
为了使5.4更具体，我们需要对寻找α_k的线性搜索法有更精确的叙述。由于(5.41b)中p_k可能不一定是下降方向，除非α_k满足一些特定条件。

我们将(5.41b)和∇f_k做内积可得到(5.42)

$$\nabla f_k^T p_k = -\|\nabla f_k\|^2 + \beta_k^{FR}\nabla f_k^T p_{k-1}$$

如果线性搜索是精确的，则α_{k-1}是f沿着p_{k-1}方向的极小子，于是我们有$\nabla f_k^T p_{k-1} = 0$，于是此时由(5.42)可推出$\nabla f_k^T p_k < 0$，即p_k确实是下降方向。如果线性搜索是非精确的，则(5.42)的第二项可

能成为主导项，则我们可能有$\nabla f_k^T p_k > 0$，此时p_k是上升方向。然而幸运的是，我们可以通过要求步长满足强Wolfe准则来避免这个问题，即(5.43)

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$$
$$\|\nabla f(x_k + \alpha_k p_k)^T p_k\| \leq -c_2 \nabla f_k^T p_k$$

其中$0 < c_1 < c_2 < \frac{1}{2}$。注意到这里我们要求$c_2 < \frac{1}{2}$，而不仅仅是之前在(3.7)里要求的$c_2 < 1$。通过下面的引理5.6，我们可以知道(5.43b)其实保证了(5.42)是负的，于是我们可以推出任意满足(5.43)的线性搜索法结果α_k都能够保证p_k始终是原函数的下降方向。

Polak-Ribère法及其变种

根据选择β_k的方法的不同，FR算法有很多变种，其中一个非常重要的变种，由Polak和Ribère提出，定义β_k如下：

$$\beta_k^{PR} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}$$

我们用(5.44)代替算法FR中的(5.41a)。根据(5.16)的正交性结论，当f是强凸二次函数且做了精确线性搜索法时，这个算法和FR是等价的。当对一个任意的非线性函数f使用非精确搜索时，这两个算法的效果是有显著区别的。数值实验说明PR算法往往更稳定而高效。

关于PR算法，有个特别值得注意的事实是 强Wolfe条件并不能保证p_k总是下降方向，假如我们如下定义β：(5.45)

$$\beta_{k+1}^{+} = \max\{\beta_{k+1}^{PR}, 0\}$$

则可以得到一个更好的算法——PR+算法，此时强Wolfe条件可以保证梯度下降性质。

在优化目标为二次函数且线性搜索精确的时候，还有很多关于β_{k+1}的选择与FR公式是一致的，如 Hestenes-Stiefel 公式，定义如下(5.46)

$$\beta_{k+1}^{HS} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k^T p_k}$$

则推导出了在理论收敛性和实践性能上均与PR算法相似的HS算法。
(5.46)可以通过要求连续搜索的方向关于*平均Hessian*在$[x_k, x_{k+1}]$上共轭，即

$$\bar{G}_k \equiv \int_0^1 [\nabla^2 f(x_k + \tau \alpha_k p_k)] \mathrm{d}\tau$$

由Taylor定理，我们有$\nabla f_{k+1} = \nabla f_k + \alpha_k \bar{G}_k p_k$，于是我们有：如果$p_{k+1} = -\nabla f_{k+1} + \beta_{k+1} p_k$，且$p_{k+1}^T \bar{G}_k p_k = 0$，则有(5.46)。

下面我们将证明，保证全局收敛性的一个条件(5.47):

$$|\beta_k| \leq \beta_k^{FR}, \quad \forall k \geq 2$$

这个条件推导出下述FR算法的修正版本，这个版本在一些应用领域的效果还不错。(5.48)

$$\beta_k = \begin{cases} -\beta_k^{FR} & \text{if } \beta_k^{PR} < -\beta_k^{FR} \\ \beta_k^{PR} & \text{if } |\beta_k^{PR}| \leq \beta_k^{FR} \\ \beta_k^{FR} & \text{if } \beta_k^{PR} > -\beta_k^{FR} \end{cases} \quad \forall k \geq 2$$

使用这种策略的算法被记为FR-PR算法。

近期也有其他一些共轭梯度法的变种形式，两种在理论和计算效果上值得引起关注的选择如下：(5.49)

$$\beta_{k+1} = \frac{\|\nabla f_{k+1}\|^2}{(\nabla f_{k+1} - \nabla f_k)^T p_k}$$

(5.50)

$$\beta_{k+1} = \left(\hat{y}_k - 2p_k \frac{\|\hat{y}_k\|^2}{\hat{y}_k^T p_k}\right)^T \frac{\nabla f_{k+1}}{\hat{y}_k^T p_k}, \quad \text{with } \hat{y}_k = \nabla f_{k+1} - \nabla f_k$$

在α_k满足Wolfe条件时，这两种做法都保证了p_k是一个下降方向。且基于这两种选择的共轭梯度法的性能似乎与PR算法差不多。

二次终止和重启动

大部分非线性共轭梯度法都与线性共轭梯度法有着紧密的联系。通常，线性搜索法中会使用二次或三次插值算法，于是当f是严格凸二次函数时，α_k的选取就是当前方向上的极小子，因此，非线性共轭梯度法会退化成线性版本，即算法5.2。

非线性共轭梯度法的另一种修正思路是每n次迭代后设(5.41a)中的$\beta_k = 0$以此来**重启动**算法，也就是说使每n步用一个最速下降法。重启动是一种周期性更新算法的方法，它会去除那些可能无益的信息。我们甚至可以给出一个很强的理论结果：这种策略能够引导出n步的二次收敛性质，即(5.51)

$$\|x_{k+n} - x^*\| = O(\|x_k - x^*\|^2)$$

稍加思考我们可以发现这个结果并不意外。假设f在解的领域内强凸二次，却不是在任何地方都是二次的，那么这种迭代方法最终会进入到这个二次区域中。当我们在某些地方重启动算法时，我们的算法可能会退化为线性共轭梯度法5.2。而且，这种方法是能在有限次迭代中停止。由于5.2的有限终止性和其他一些性质都要求初始方向是负梯度方向，因此，重启动是非常重要的一种手段。

即使f在解附近不是精确二次的，Taylor 定理告诉我们如果它是光滑的，呢么它在解的附近也是近似于二次的。也就是说，我们并不期望算法能在重启动后n步终止，但是像(5.51)表述的一样，算法阿是逐渐收敛到真解的。

虽然从理论上看(5.51)是非常有趣的，但是实际上并不是非常有用，因为我们只在处理大规模问题的时候才会考虑非线性共轭梯度法，对于这样的问题，我们也许根本用不上重启动，因为近似解极有可能在n步内得到。因此，NCG有时候是不带重启动的，或者用一些其他指标去进行重启动。最常用的策略就是利用(5.16)，即梯度在f是二次函数时正交，因此，我们可以让算法在两个梯度极不正交的时候重启动，即(5.52)

$$\frac{|\nabla f_k^T \nabla f_{k-1}|}{\|\nabla f_k\|^2} \geq \nu$$

常用的$\nu = 0.1$。

我们也可以利用(5.45)作为我们的重启动条件，因为当β_k^{PR}为负数时，p_{k+1}需要取最速下降方向。与(5.52)不同的是，这种做法很少重启动，因为β_k^{PR}大部分时间都是正数。

Flecher-Reeves法的效果

下面我们对FR算法做一些分析，首先证明它的全局收敛性，然后解释它不够高效的一些原因。

这里我们要求线性搜索的方向是下降方向。同时，$\mathcal{L} = \{x : f(x) \leq f(x_0)\}$是有界的，$f$二次可微，于是，前述条件等价于$\alpha_k$满足强Wolfe条件

Lemma 5.6

> 假设5.4中的α_k满足$0 < c_2 < 1$的强Wofle条件(5.43)，那么，这个方法能够产生满足下述条件的下降方向p_k:(5.53)
>
> $$-\frac{1}{1-c_2} \leq \frac{\nabla f_k^T p_k}{\|p_k\|^2} \leq \frac{2c_2 - 1}{1 - c_2}, \quad \forall k = 0, 1, \ldots$$

证明：
注意到$t(\xi) \overset{def}{=} (2\xi - 1)/(1 - \xi)$在区间$[0, \frac{1}{2}]$上单增，且$t(0) = -1, t(\frac{1}{2}) = 0$，因此由于$c_2 \in (0, \frac{1}{2})$，我们有(5.54)

$$-1 < \frac{2c_2 - 1}{1 - c_2} < 0$$

结合(5.53)我们便得到p_k是下降方向。

而(5.53)的证明是借助Wolfe条件，利用归纳法完成的。

上述证明只用到了Wolfe条件(5.43b)，而(5.43a)则是用于保证全局收敛性的。

引理5.6也可以用来说明FR方法的缺点。如果一个算法产生了一个不好的方向和一个很小的步长，那么下一步很可能也很糟糕，如同Ch.3中的做法一样，我们定义(5.56):

$$\cos\theta_k = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\| \|p_k\|}$$

假设p_k是一个不好的方向，即几乎垂直于负梯度方向，$\cos\theta_k \approx 0$。那么通过在(5.53)两端乘以$\|\nabla f_k\|/\|p_k\|$，并利用表达式(5.56)，我们有(5.57)

$$\frac{1-2c_2}{1-c_2}\frac{\|\nabla f_k\|}{\|p_k\|} \leq \cos\theta_k \leq \frac{1}{1-c_2}\frac{\|\nabla f_k\|}{\|p_k\|}, \quad \forall k = 0, 1, \ldots$$

那么利用上述不等式，我们可以推出$\cos\theta_k \approx 0$当且仅当

$$\|\nabla f_k\| \ll \|p_k\|$$

由于p_k几乎垂直于梯度，于是x_k到x_{k+1}的变化很小，即，$x_{k+1} \approx x_k$，那么$\nabla f_{k+1} \approx \nabla f_k$，于是(5.58)

$$\beta_{k+1}^{FR} \approx 1$$

总之，$p_{k+1} \approx p_k$，于是新的搜索方向还是很糟糕。于是如果在某些迭代步上$\cos\theta_k \approx 0$且步长很小，那么将产生很多低效率的迭代。

而PR算法的行为就很不一样，假设p_k满足$\cos\theta_k \approx 0$且子步长很小，于是由PR算法可知$\nabla f_k \approx \nabla f_{k+1}$，此时$\beta_{k+1}^{PR} \approx 0$，再由(5.41b)，$p_{k+1}$将接近于负提地方性。因此相当于PR方法在遭遇了一次不好的迭代以后做了重启动。而PR+和HS算法里也有一样的机制。

在FR算法的实际使用中，这个问题是一个很关键的问题，但是，我们也可以通过加入周期性的重启动，让FR算法离开这种低效循环，从而提高效率。因此FR算法在使用的时候一定要加上重启动策略。

全局收敛性

跟线性的情况大相径庭，非线性共轭梯度法的收敛性质是很特殊甚至奇怪的。下面我们展示一些FR算法和PR算法的主要结果。

本节我们对目标函数做如下假设：(假设5.1)

1. 水平集$\mathcal{L} = \{x | f(x) \leq f(x_0)\}$有界
2. 在水平集$\mathcal{L}$的开领域$\mathcal{N}$内，目标函数Lipschiz连续可微

从上述假设，可推出，存在$\bar{\gamma}$ s.t.(5.59)

$$\|\nabla f(x)\| \leq \bar{\gamma}, \quad \text{for all} x \in \mathcal{L}$$

我们的主要分析手段是Zoutendijk定理(Theorem 3.2)，在假设5.1下，线性搜索法能够给出如下结果:(5.60)

$$\sum_{k=0}^{\infty} \cos^2\theta_k \|\nabla f_k\|^2 < \infty$$

我们可以利用这个结果证明带周期重启动的算法的全局收敛性，记$k_1, k_2, \ldots$是那些重启动的迭代步，则:(5.61)

$$\sum_{k=k_1,k_2,\dots} \|\nabla f_k\|^2 < \infty$$

如果我们允许两次重启间的迭代次数不超过$\bar{n}$次，且$\{f_k\}$是个无穷序列，则从(5.61)我们有$\lim_{j\to\infty} \|\nabla f_{k_j}\| = 0$，即存在梯度趋向0的子序列，也即(5.62)

$$\liminf_{k\to\infty} \|\nabla f_k\| = 0$$

这个结果对于所有本章讨论的算法加上重启动机制后都适用。

当然，更有趣的研究是研究那些不使用重启动机制的算法的收敛性。因为对于大规模($n > 1000$)问题，我们希望通过远小于n次的迭代找到解——这种情况通常是不触发重启动的。然而对于这种问题的研究有一些特殊的结论。

我们可以基于引理5.6和Zoutendijk定理去证明FR方法的全局收敛性，但我们无法证明梯度的极限是0，下面的结果保证了下极限是0(5.63)

Theorem 5.7

在假设5.1的条件下，如果算法5.4使用满足(5.43)的线性搜索法，其中$0 < c_1 < c_2 < \frac{1}{2}$，则有(5.63)

$$\liminf_{k\to\infty} \|\nabla f_k\| = 0$$

证明：
我们主要使用反证法说明这个结论。首先假设(5.63)的逆命题成立，即
假设存在$\gamma > 0$ s.t.

$$\|\nabla f_k\| \geq \gamma, \forall \text{ large } k$$

于是将(5.57)左式代入Zoutendijk条件(5.60)，我们有(5.65)

$$\sum_{k=0}^{\infty} \frac{\|\nabla f_k\|^4}{\|p_k\|^2} < \infty$$

利用(5.43b)和(5.53)，我们有(5.66)

$$|\nabla f_k^T p_{k-1}| \leq -c_2 \nabla f_{k-1}^T p_{k-1} \leq \frac{c_2}{1-c_2} \|\nabla f_{k-1}\|^2$$

因此，由(5.41b)和(5.41a)我们有

$$\begin{aligned}
\|p_k\|^2 \leq & \|\nabla f_k\|^2 + 2\beta_k^{FR} \|\nabla f_k^T p_{k-1}\| + (\beta_k^{FR})^2 \|p_{k-1}\|^2 \\
\leq & \|\nabla f_k\|^2 + \frac{2c_2}{1-c_2} \beta_k^{FR} \|\nabla f_{k-1}\|^2 + (\beta_k^{FR})^2 \|p_{k-1}\|^2 \\
\leq & \left(\frac{1+c_2}{1-c_2}\right) \|\nabla f_k\|^2 + (\beta_k^{FR})^2 \|p_{k-1}\|^2
\end{aligned}$$

利用这个结果，定义$c_3 \overset{def}{=} (1+c_2)/(1-c_2) \geq 1$，我们有(5.67)

$$\begin{aligned}\|p_k\|^2 \leq & c_3\|\nabla f_k\|^2 + (\beta_k^{FR})^2(c_3\|\nabla f_{k-1}\|^2 + (\beta_k^{FR})^2(\cdots)) \\ = & c_3\|\nabla f_k\|^4 \sum_{j=0}^{k} \|\nabla f_j\|^{-2}\end{aligned}$$

这是因为

$$(\beta_k^{FR})^2(\beta_{k-1}^{FR})^2 \cdots (\beta_{k-i}^{FR})^2 = \frac{\|\nabla f_k\|^4}{\|\nabla f_{k-i-1}\|^4}$$

同时 $p_0 = -\nabla f_0$，通过使用(5.59)和(5.64)，我们有(5.68)

$$\|p_k\|^2 \leq \frac{c_3\bar{\gamma}^4}{\gamma^2}k$$

这个结果说明，存在某些正常数γ_4(5.69)

$$\sum_{k=1}^{\infty} \frac{1}{\|p_k\|^2} \geq \gamma_4 \sum_{k=1}^{\infty} \frac{1}{k}$$

另一方面，利用(5.64)和(5.65)，有(5.70)

$$\sum_{k=1}^{\infty} \frac{1}{\|p_k\|^2} < \infty$$

然而，如果我们吧这个不等式与(5.69)联立，则有$\sum_{k=1}^{\infty} 1/k < \infty$，这是不对的。因此导出矛盾。

这个全局收敛性可以拓展到任何β_k满足(5.47)的算法，也包括FR-PR算法。

一般来说，如果我们可以说明存在$c_4, c_5 > 0$ s.t.

$$\cos\theta_k \geq c_4 \frac{\|\nabla f_k\|}{\|p_k\|}, \quad \frac{\|\nabla f_k\|}{\|p_k\|} \geq c_5 > 0, \quad k = 1, 2, \ldots$$

则由(5.60), 我们可以得到

$$\lim_{k\to\infty} \|\nabla f_k\| = 0$$

事实上，对于PR算法，我们可以假设f是强凸函数，且线性搜索精确时证明这个结果。

对于更一般的(非凸)函数问题，PR算法是没有类似于定理5.7的结果的。这个结果是很令人以外的，因为从实践效果上来说PR算法是优于FR算法的。而下面的结果则说明了PR算法可能在解附近无穷打转，即使用了最理想的线性搜索法(即返回一维函数的稳定点。)

Theorem 5.8

对于使用了理想线性搜索法的PR算法(5.44)，存在一个二次连续可微函数$f:R^3\rightarrow R$，和一个初值x_0使得$\{\|\nabla f_k\|\}$有界远离0。

这个证明非常复杂。它用了一中解释存在性而不是显式构造的证明方式。
因此，这个定理的证明过程中要求其中一些搜索方向几乎是另一些搜索方向的相反数，在理想线性搜索中，这只会在$beta_k<0$时发生，因此这个分析建议使用PR+算法(5.45)。
我们之前提到了只要对Wolfe条件做一些改动，就能够得到PR+算法的全局收敛性，然而(5.49)和(5.50)则不需要对Wolfe条件做任何改变就可以得到全局收敛的结果。

数值效果和拓展

PR和PR+算法并不总是比FR高效许多，而且它们还需要储存更多的向量，但是我们还是推荐使用PR、PR+、FR-PR或者是其它基于(5.49)和(5.50)的共轭梯度法。

Powell证明了FR有时候甚至比最速下降法还慢。

Crowder 和 Wolfe 证明了收敛率是线性的。Powell证明要么有限步停止，要么线性收敛。