CONJUGATE GRADIENT METHOD AND ITS EXTENSIONS

Zewdie Tibebu

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE
DEPARTMENT OF MATHEMATICS

A Project submitted in partial fulfillment of the requirement of the degree of master of science in mathematics

Advisor: Berhanu Guta(PhD.)

May, 2018

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY
DEPARTMENT OF MATHEMATICS

The undersigned hereby certify that they have read and recommend to the department of mathematics for acceptance of this project entitled **"Conjugate Gradient Method and its Extension"** by **Zewdie Tibebu** in partial fulfillment of the requirements for the degree of Master of Science in Optimization.

Advisor: Dr. Berhanu Guta

Signature:_____

Date_____

Examiner 1: Dr._____

Signature:_____

Date_____

Examiner 2: Dr. _____

Signature:_____

Date_____

ADDIS ABABA UNIVERSITY
DEPARTMENT OF MATHEMATICS

Author: Zewdie Tibebu

Title: **Conjugate Gradient Method and its Extension**

Department: Mathematics

Degree: M.Sc.

Convocation: May

Year: 2018

Zewdie Tibebu:
Signature:_____

Date_____

Contents

Acknowledgement

I am grateful to the Almighty God for He is my source of inspiration. I am specially grateful to my advisor Dr. Berhanu Guta for guiding, giving me supporting materials and directing the success of this project. And his great assistance, contribution, useful suggestion and constructive comments on this excellence advising and limitless effort in encouraging me in my work, correcting and giving comments by devoting his time. I greatly appreciate him not only for unlimited constructive advises in every aspect of my project work, but also with his professional and personal ethic, that should be an icon for others and I am specially grateful to my advisor.

Next, I would like to thanks my family specially my father Tibebu Gebeyehu and my mother Mulu Mekuonen and my brother Belaynew Tibebu and Mulugeta Adugnaw for your financial support and helpful praise is unforgettable in my life. I thank you all for the love you have shown me through my entire studies, specially my heartfelt gratitude and appreciation goes to my Lovely friend Abaynew Zemene for he continuing love, motivation, encouragement, and constructive advice throughout this long and challenging process.

I would like to thanks to all respective organizations, Debre tabor university for sponsoring and giving a permission for this MSc study Adiss Ababa university, Optimization program. Also I want to thanks Addis Ababa university and all staff members of Mathematics Department at Addis Ababa university for helping me during my study.

Finally, I would like to thank my classmates your appreciation and comment really valuable.

Abstract

In this project we investigate conjugate gradient method and its extension to solve unconstrained minimization problems. There are two important methods for solving linear equations and nonlinear optimization problems. The performance of the linear conjugate gradient method is tied to the distribution of the eigenvalues of the coefficient matrix. Nonlinear conjugate gradient method is used for solving large-scale nonlinear optimization problems and has wide applications in many fields. It is also discussed how to use the result to obtain the convergence of the famous Fletcher-Reeves, and Polak-Ribiere conjugate gradient methods. And comparisons are made among the algorithms of the steepest descent method, Newton's method and conjugate gradient method for quadratic and nonquadratic problems.

Introduction

We consider the unconstrained minimization problem

$$\min_{x \in \Re^n} f(x), \tag{1}$$

where $f : \Re^n \to \Re$ is continuously differentiable. Iterative methods for unconstrained Nonlinear programming are generally grouped into two classes: Line search methods and Trust-region methods both generate steps with the help of a quadratic model of the objective function, but they use this model in different ways. Line search methods use it to generate a search direction, and then focus their efforts on finding a suitable step length α along this direction. Trust-region methods define a region around the current iterate within which they trust the model to be an adequate representation of the objective function, and then choose the step to be the approximate minimizer of the model in this region. But we use line search methods to solve quadratic and nonquadratic model of the objective function. There are many line search methods in numerical optimization. Some of them are steepest descent method, Newton's method and conjugate gradient method etc.

The focus of this project is Conjugate Gradient Method and its Extension. Conjugate gradient methods there are two important methods. First, linear conjugate gradient methods and second nonlinear conjugate gradient methods.

The linear conjugate gradient method was introduced by Hestenes and Stiefel[1] in 1950's for solving a symmetric positive definite linear system $Ax = b$, where $A \in \Re^{n \times n}$ and $b \in \Re^n$. Geometrical interpretation of the linear conjugate gradient method can be founded in Nocedal[1]. The equivalence of the linear system to the minimization problem of $f(x) = \frac{1}{2}x^T Ax - b^T x + c$, motivated Fletcher and Reeves [1] to extend the linear conjugate gradient method for nonlinear optimization.

In this project we introduce the conjugate gradient method which is one between the steepest descent method and the Newton method. The conjugate gradient method deflects the direction of the steepest descent method by adding to it a positive multiple of the direction used in the last step. This method only requires the first-order derivatives but overcomes the steepest descent methods shortcoming of slow convergence. At the same time, the method need not save and compute the second-order derivatives which are needed by Newton method. In particular, since it does not require the Hessian matrix or its approximation, it is widely used to solve large scale optimization problems. Conjugate gradient methods are iterative methods and at the k^{th} iteration, the form is given by

$$x_{k+1} = x_k + \alpha_k p_k, \tag{2}$$

where $\alpha_k > 0$ is a positive step size and p_k is a search direction. Search directions are usually defined by

$$p_{k+1} = \begin{cases} -r_{k+1}, & \text{for } \; k = 0 \\ -r_{k+1} + \beta_k p_k, & \text{for } \; k \geq 1 \end{cases}, \tag{3}$$

where r_{k+1} denotes $\nabla f(x_{k+1})$ at the point x_{k+1} and $\beta_k \in \Re$ is a scalar parameter, which characterizes conjugate gradient methods. It is known equation (2) and (3) that only the stepsize α_k and the parameter β_k remain to be determined in the definition of conjugate gradient methods. In the case that $f(x)$ is a convex quadratic, the choice of β_k should be such that the method (2)-(3) reduces to the linear conjugate gradient method if the line search is exact, namely,

$$\alpha_k = \arg\min_{\alpha>0} f(x_k + \alpha p_k). \tag{4}$$

For nonlinear functions, however, different formulae for the parameter β_k result in different conjugate gradient methods and their properties can be significantly different. To differentiate the linear conjugate gradient method, sometimes we call the conjugate gradient method for unconstrained optimization by nonlinear conjugate gradient method. Meanwhile, the parameter β_k is called conjugate gradient parameter.

In this project to provide a perspective view on the methods from the angle of descent property and global convergence. Since the exact line search is usually expensive and impractical, the strong Wolfe line search is often considered in the implementation of nonlinear conjugate gradient methods. It aims to find a stepsize satisfying the strong Wolfe conditions

$$f(x_k) - f(x_k + \alpha_k p_k) \geq -c_1 \alpha_k r_k^T p_k, \tag{5}$$

$$|f(x_k + \alpha_k p_k)^T p_k| \leq -c_2 \nabla f_k^T p_k \tag{6}$$

where $0 < c_1 < c_2 < 1$.

On the other hand, for a general nonlinear function, one may be satisfied with a stepsize satisfying the standard Wolfe conditions, namely, (6) and

$$f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k \tag{7}$$

where again $0 < c_1 < c_2 < 1$. A requirement for an optimization method to use the above line searches is that, the search direction p_k must have the descent property, namely,

$$r_k^T p_k < 0. \tag{8}$$

For conjugate gradient methods, by multiplying (3) with r_k^T , we have

$$r_k^T p_k = -\|r_k\|^2 + \beta_{k-1} r_k^T p_{k-1}. \tag{9}$$

Thus if the line search is exact, we have $r_k^T p_k = -\|r_k\|^2$ since $r_k^T p_{k-1} = 0$. Consequently, p_k is descent provided $r_k \neq 0$. In this project we say that, a conjugate gradient method is descent if (9) holds for all k, and is sufficient descent if the sufficient descent condition

$$r_k^T p_k \leq -c\|r_k\|^2 \tag{10}$$

holds for all k and some constant $c > 0$.

The project paper has been organized in to four chapters. In chapter one we introduce the baseline of the study or dealing with the mathematical background of the project, definition of basic terms, Algorithms and Theorems. In chapter two, we incorporate Linear conjugate gradient method, which are more relevant for further investigation of the study. In chapter three, we describe the Nonlinear conjugate gradient method and the global convergence. In chapter Four, we describe the Numerical implementation and the discussion of results. Finally, the conclusions from the output and recommendation are presented in section.

Chapter 1

Preliminaries

1.1 Iterative Methods

Definition 1.1.1. *An iterative method is said to be*

i. locally convergent if the sequence of iterates generated by the method converges to a solution when the initial approximation is already close enough to the solution.

ii. globally convergent if the sequence of iterates generated by the method converges for any initial approximation.

1.2 Optimality Conditions for Unconstrained Optimization

Definition 1.2.1. *A point x^* is called a local minimizer if there exists $\delta > 0$ such that $f(x^*) \leq f(x)$ for all $x \in \Re^n$ satisfying $\|x - x^*\| < \delta$. A point x^* is called a strict local minimizer if there exists $\delta > 0$ such that $f(x^*) < f(x)$ for all $x \in \Re^n$ with $x \neq x^*$ and $\|x - x^*\| < \delta$.*

Definition 1.2.2. *A point x^* is called a global minimizer if $f(x^*) \leq f(x)$ for all $x \in \Re^n$. A point x^* is called a strict global minimizer if $f(x^*) < f(x)$ for all $x \in \Re^n$ with $x \neq x^*$.*

Definition 1.2.3. ***descent direction:*** *Consider a function $f : \Re^n \longrightarrow \Re$ and a point $x_k \in \Re^n$. A nonzero vector $p_k \in R^n$ is called a descent direction of f at x_k if and only if there is $\alpha > 0$ such that*

$$f(x_k + tp_k) < f(x_k),$$

for all $t \in (0, \alpha)$.

Let us assume that f is continuously differentiable at $x \in \Re^n$. Then

$$\nabla f(x)^T p < 0 \tag{1.1}$$

is **sufficient** to show that $p \in \Re^n$ is a descent direction of f in x.

Theorem 1.1. *Consider a function $f : \Re^n \longrightarrow \Re$ and a point $x_k \in \Re^n$.*
If $\nabla f(x_k)^T p_k < 0$, then p_k is a descent direction of f at x_k.

Theorem 1.2. *(**Taylors Theorem**) Suppose that $f : \Re^n \to \Re$ is continuously differentiable, and that $p \in \Re^n$. Then we have that*

$$f(x+p) = f(x) + \nabla f(x+tp)^T p,$$

for some $t \in (0,1)$. Moreover, if f is twice continuously differentiable, we have that

$$\nabla f(x+p) = \nabla f(x) + \int_0^1 \nabla^2 f(x+tp)p dt, \tag{1.2}$$

and that

$$f(x+p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x+tp)p, \quad for\ some\ t \in (0,1).$$

if and only if p is a descent direction of f at x.

Definition 1.2.4. ***Convex Functions:-*** *A function $f : \Re^n \to \Re$ is said to be convex if for any $x_1, x_2 \in \Re^n$ and any $\lambda \in [0,1]$,*

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2).$$

.f is strictly convex if for every distinct $x_1, x_2 \in \Re^n$, and $\lambda \in (0,1)$,

$$f(\lambda x_1 + (1-\lambda)x_2) < \lambda f(x_1) + (1-\lambda) f(x_2).$$

1.3 Rate of Convergence

Definition 1.3.1. *Let $\{x_k\}$ be a sequence in $\Re^n$ that converges to x^*. The rate of convergence is said be*

(a) linear if there is a constant $r \in (0,1)$ and $k_0 \geq 0$ such that

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \leq r \quad for\ all \quad k \geq k_0$$

(b) supper linear if $\lim_{k \longrightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0$

(c) quadratic if there is a constant $M > 0$ and $k_0 \geq 0$ such that

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} \leq M \quad for\ all \quad k \geq k_0$$

Definition 1.3.2. ***Positive Definite Matrices:*** *Let A be any $n \times n$ symmetric matrix. The matrix A is called positive definite if all of its eigenvalues are strictly positive. This is denoted by $A > 0 \Leftrightarrow x^T A x > 0 \qquad$ for all $x \in \Re^n, x \neq 0$.*

1.4 Line search methods

Definition 1.4.1. *Line search, also called one-dimensional search, refers to an optimization procedure for univariable functions. It is the base of multivariable optimization. As stated before, in multivariable optimization algorithms, for given x_k, the iterative scheme is*

$$x_{k+1} = x_k + \alpha_k p_k \tag{1.3}$$

The key is to find the direction vector p_k and a suitable step size α_k. Let

$$\phi(\alpha) = f(x_k + \alpha p_k).$$

So, the problem that departs from x_k and finds a step size in the direction p_k such that $\phi(\alpha_k) < \phi(0)$ is just line search about α.

▪ *If we find α_k such that the objective function in the direction p_k is minimized,*

$$i.e., \quad f(x_k + \alpha_k p_k) = \min_{\alpha>0} f(x_k + \alpha p_k),$$

or

$$\phi(\alpha_k) = \min_{\alpha>0} \phi(\alpha),$$

*such a line search is called **exact line search** or optimal line search, and α_k is called optimal step size.*

Algorithm 1.1. ***(Exact Line search Method)***

Step 1 : *Start with an initial guess* $x_1 \in \Re^n$, $(k = 1)$
Step 2 : *At iterate point x_k, check whether x_k satisfy optimality condition*
▪ *If Yes Stop. (x_k is optimal solution).*
▪ *If No, GOTO Step* 3.
Step 3 : *Determine a descent direction of f at x_k. Let this be p_k.*
▪ *Find (local) minimizer of f along p_k, i.e., let $\phi(t) = f(x_k + tp_k)$, $t \geq 0$, and (i) determine IoU $[a, b]$ of $\phi(t)$, (% interval of uncertainty)*
(ii) solve the $1-d$ problem : $\min \phi(t)$ s.t. $t \in [a, b]$ %(Exact)
If the outcome of this $1-d$ problem is t^*, then let $t_k = t^*$ and $x_{k+1} = x_k + t_k p_k$ is the next iterate point. $(k := k + 1)$
Repeat Step 3 *until a termination (optimality) condition is met.*

▪ *If we choose α_k such that the objective function has acceptable descent amount, i.e., such that the descent $f(x_k) - f(x_k + \alpha_k p_k) > 0$ is acceptable by users, such a line search is called **inexact line search**, or approximate line search, or acceptable line search.*

Definition 1.4.2. ***Wolfe's conditions:*** *Let c_1 and c_2 be two constants such that $0 < c_1 < c_2 < 1$. We say that the step length α_k satisfy the Wolfe conditions if α_k satisfy:*

1. ***sufficient decrease:*** $f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$

2. ***curvature condition:*** $\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k$.

Definition 1.4.3. ***Strong Wolfe conditions****: Let c_1 and c_2 be two constants such that $0 < c_1 < c_2 < 1$. We say that the step length α_k satisfy the strong Wolfe conditions if α_k satisfy:*

1. ***sufficient decrease:*** $f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$

2. ***curvature condition:*** $|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq c_2 \nabla f_k^T p_k$.

Algorithm 1.2. ***(Inexact line search Algorithm)***
At iteration t_k and descent direction p_k,
▪ *Choose a sufficiently small (not too small)$\delta > 0$ such that*

$$f(x_k + \delta p_k) < f(x_k) \quad and \quad c_2 \in (0.1, 1), \quad say \quad c_2 = 0.4,$$

▪ *Let $t_k = \delta$ (% Initialization)*
▪ *While $\nabla f(x_k + t_k p_k)^T p_k \leq c_2 \nabla f_k^T p_k$*

$$t_k = t_k + \delta; (\%Increase \quad t_k \quad by \quad \delta)$$

end (% end while)
▪ *The last t_k at termination of the while-loop satisfies the Wolfe condition.*

Lemma 1.1. *Suppose that $f : \Re^n \longrightarrow \Re$ is continuously differentiable. Let p_k be a descent direction at x_k, and assume that f is bounded below along the ray $\{x_k + \alpha p_k | \alpha > 0\}$. Then if $0 < c_1 < c_2 < 1$, there exist intervals of step lengths satisfying the above Wolfe conditions and the strong Wolfe conditions.*

Theorem 1.3. ***(Zoutendijk Theorem)***
Consider any iteration of the form (1.3), where p_k is a descent direction and α_k satisfies the Wolfe conditions. Suppose that f is bounded below in $\Re^n$ and that f is continuously differentiable in an open set $\aleph$ containing the level set $L = x : f(x) \leq f(x_0)$, where x_0 is the starting point of the iteration. Assume also that the gradient ∇f is Lipschitz continuous on $\aleph$, that is, there exists a constant $L > 0$ such that

$$\|\nabla f(x) - \nabla f(\tilde{x})\| \leq L\|x - \tilde{x}\|, \quad for\ all\ x, \tilde{x} \in \aleph.$$

Then

$$\sum_{k \geq 0} \cos^2 \theta_k \|\nabla f_k\|^2 < \infty, \tag{1.4}$$

where θ_k is the angle between $-\nabla f_k$ and p_k at each k.

1.5 Steepest Descent method

Definition 1.5.1. *The steepest descent method is one of the simplest and the most fundamental minimization methods for unconstrained optimization. Since it uses the negative gradient as its descent direction, it is also called the gradient method.*

Suppose that $f(x)$ is continuously differentiable near x_k, and the gradient $g_k = \nabla f(x_k) = 0$. From the Taylor expansion

$$f(x) = f(x_k) + (x - x_k)^T g_k + o(\|x - x_k\|),$$

we know that, if we write $x - x_k = \alpha p_k$, then the direction p_k satisfying $p_k^T g_k < 0$ is called a descent direction that is such that $f(x) < f(x_k)$. Fixing α, it follows that the smaller the value $p_k^T g_k$ (i.e., the larger the value $|p_k^T g_k|$) is, the faster the function value decreases. By the Cauchy-Schwartz inequality

$$|p_k^T g_k| \leq \|p_k\| \|g_k\|,$$

we have that the value $p_k^T g_k$ is the smallest if and only if $p_k = -g_k$. Therefore $-g_k$ is the steepest descent direction. The iterative scheme of the steepest descent method is

$$x_{k+1} = x_k - \alpha_k g_k.$$

In the following we give the algorithm.

Algorithm 1.3. *(The Steepest Descent Method)*
Step 1.Choose small $\epsilon > 0$ and initial guess x_0, $k = 0$.
Step2. while $\|g_k\| > \epsilon$,

- *let* $p_k = -g_k$.
- *Get t_k that minimizes $\phi(t) = f(x_k + tp_k)$,* $t \geq 0$
- $x_{k+1} = x_k + t_k p_k$.

Step 3. $k = k + 1$.
end(%end while)

Properties of Steepest Descent Method

1. *It is convergent (if the function has a minima) starting from any x_0.*
2. *However, the convergence is slow (linear rate) due to its zigzagging nature: $\nabla f(x_{k+1}) \perp p_k$ at every consecutive iterate points.*
3. *So, it is rather faster with inexact line search (α_k satisfying Wolfe Conditions).*

1.6 Newton's Method

Newtons Method is a line search method for multidimensional minimization problem. That is, given a current iterate point $x_k \in \Re^n$, the next iterate point x_{k+1} is the point that

minimizes the quadratic approximation of f around x_k. This approximating function is
$q(x) = f(x_k) + \nabla f(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)$
So, at x_{k+1} the gradient of q should be zero.
Thus, since

$$\begin{aligned} \nabla q(x) &= -\nabla f(x_k) + \nabla^2 f(x_k)(x - x_k), \\ \nabla q(x_{k+1}) = 0 &\Rightarrow \nabla f(x_k) + \nabla^2 f(x_k)(x - x_k) = 0 \\ &\Rightarrow \nabla^2 f(x_k)(x_{k+1} - x_k) = -\nabla f(x_k) \\ &\Rightarrow x_{k+1} = x_k [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) \end{aligned}$$

Draw back of Newton's method
When the starting point is far away from the solution, it is not sure that $\nabla^2 f(x_k)$ is positive definite and Newtons direction p_k is a descent direction. Hence the convergence is not guaranteed.

Algorithm 1.4. *(Newton's Method)*
Input: $f(x)$, *gradient* $\nabla f(x)$, *Hessian matrix* $H(x)$, *tolerance* > 0
1. Choose a suitable initial guess $x_1 \in \Re^n$
$k = 1$,
If $\|\nabla f(x_k)\| < \epsilon$, *Stop (* $x^* = x_k$ *is an approximate solution)*
Else
Solve the system of linear equations
$\nabla^2 f(x^k) X = \nabla f(x_k)$ *(*)*
$x_{k+1} = x_k + X$ *, where* X *is the solution of (*).*
Set $k = k + 1$, *and repeat from step 2*
end (%end if)

1.7 Krylov subspaces

Definition 1.7.1. *Let* S^n *be space of real symmetric matrices of order* $n, r \in \Re^n$ *and* $A \in S^n$. *The order* k *Krylov subspace generated by* A *and* r, *and denoted by* $K(A, r, k)$ *is the subspace:*

$$K(A, r, k) = span(r_0, Ar_0, A^2 r_0, ..., A^{k-1} r_0).$$

We define the Krylov sequence $x_1, x_2, ...$ *as*

$$x_k = \arg\min_{x \in K_k} f(x) = \arg\min_{x \in K_k} \|x - x\|_A^2.$$

Definition 1.7.2. *A quadratic form* $f : \Re^n \longrightarrow \Re$ *is a function*

$$f(x) = x^T Ax + b^T x + c,$$

where A is an $n \times n$ real matrix x and b are vectors, and c is a scalar constant. There is no loss of generality in assuming A to be symmetric, that is, $A = A^T$.
The gradient of a quadratic form is defined to be

$$\nabla f(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(x) \\ \frac{\partial}{\partial x_2} f(x) \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial}{\partial x_n} f(x) \end{bmatrix}$$

Definition 1.7.3. *Let V be a vector space and $p_1, ..., p_n \in V$. Then the set $p_1, ..., p_n$ is a spanning set for V if span$\{p_1, ..., p_n\} = V$. We also say that V is generated or spanned by $\{p_1, ..., p_n\}$.*

Definition 1.7.4. ***Orthogonal Vectors:*** *Two vectors p and $q$are said to be* ***orthogonal*** *provided their dot product is zero:*

$$p \cdot q = 0.$$

Definition 1.7.5. ***Orthonormal Matrices:*** *A square matrix Q is said to be orthonormal if*

$$Q^T Q = I$$

This immediately reveals that

$$QQ^T = I \quad and \quad Q^T = Q^{-1}.$$

Definition 1.7.6. ***Hessian matrix:*** *$n \times n$ matrix of second-order partial derivatives, ordered as follows*

$$H[f(x_1, x_2, ..., x_n)] = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_2 \partial x_1} \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_1} \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Chapter 2

Linear Conjugate Gradient Methods

2.1 Conjugate Direction Methods

The class of conjugate direction methods can be viewed as being intermediate between the method of steepest descent and Newton's method. The conjugate direction methods have the following properties:

1. Solve quadratics of n variables in n steps,
2. The usual implementation, the conjugate gradient algorithm, requires no Hessian matrix evaluations,
3. No matrix inversion and no storage of an $n \times n$ matrix required.

The conjugate direction methods typically perform better than the method of steepest descent, but not as well as Newton's method. As we saw from the method of steepest descent and Newton's method, the crucial factor in the efficiency of an iterative search method is the direction of search at each iteration[2].

Conjugate direction methods invariably are invented and analyzed for the purely quadratic problem

$$\min \frac{1}{2} x^T A x - b^T x + c, \tag{2.1}$$

where A is an $n \times n$ symmetric positive definite matrix. The techniques once worked out for this problem are then extended, by approximation, to more general problems, it is being argued that, since near the solution point every problem is approximately quadratic, convergence behavior is similar to that for the pure quadratic situation.

The area of conjugate direction algorithms has been one of great creativity in the nonlinear programming field, illustrating that detailed analysis of the pure quadratic problem can lead to significant practical advances. Indeed, conjugate direction methods, especially the method of conjugate gradients, have proved to be extremely effective in dealing with general

objective functions and are considered among the best general purpose methods. One of the main properties of the conjugate gradient method is that its directions are conjugate.

Definition 2.1.1. *(a) Two distinct nonzero vectors p_1 and p_2 are said to be conjugate with respect to a real symmetric matrix A, if*

$$p_1^T A p_2 = 0.$$

(b) A finite set of distinct nonzero vectors $\{p_0, p_1, ..., p_k\}$ is said to be conjugate with respect to a real symmetric matrix A, if

$$p_i^T A p_j = 0 \quad for\ all \quad i \neq j. \tag{2.2}$$

If $A = I_n$, where I_n is the $n \times n$ identity matrix, then equation (2.2) can be expressed as $p_i^T A p_j = p_i^T I_n p_j = p_i^T p_j = 0 \quad for \quad i \neq j$. This is the well known condition for orthogonality between vectors p_i and p_j and, in effect, conjugacy is a generalization of orthogonality. If $p_j \quad for \quad j = 0, 1, ..., k$ are eigenvectors of A then $Ap_j = \lambda j p_j$ where the λ_j are the eigenvalues of A. Hence, we have $p_i^T A p_j = \lambda_j p_i^T p_j = 0 \quad for \quad i \neq j$ since p_i and $p_j \quad for \quad i \neq j$ are orthogonal.

Theorem 2.1. *If A is symmetric positive definite and the vectors $\{p_0, p_1, ..., p_{n-1}\}$ are A-orthogonal to each other, then they are linearly independent.*

Proof. Let $\alpha_0, ..., \alpha_k$ be scalars such that

$$\alpha_0 p_0 + ... + \alpha_k p_k = 0.$$

premultiplying the above equality by $p_i^T A, \quad 0 \leq i \leq k$, yields

$$\alpha_i p_i^T A p_i = 0, \quad for \quad all \quad i = 0, 1, ..., k.$$

But $p_i^T A p_i > 0$ given by the definition of symmetric positive definiteness of $A = A^T$ and $p_i \neq 0$, hence $\alpha_i = 0$ for $\quad i = 0, ..., k$.
Therefore, $p_0, p_1, ..., p_k, k \leq n - 1$ are linearly independent. □

The use of conjugate directions in the process of optimization can be demonstrated by considering the quadratic problem. Equation (2.1) where $c = f(0), r$ is the gradient of $f(x)$ at $x = 0$, and A is symmetric and positive definite. The gradient of $f(x)$ at any point can be deduced as

$$\nabla f(x) = Ax - b = r(x). \tag{2.3}$$

At the minimizer x^* of $f(x), \ r = 0$ and thus

$$Ax^* = b \tag{2.4}$$

If $\{p_0, p_1, ..., p_{n-1}\}$ are distinct conjugate directions in $\Re^n$, then they form a basis of $\Re^n$ since they are linearly independent and span the $\Re^n$ space. This means that all possible vectors in $\Re^n$ can be expressed as linear combinations of directions $\{p_0, p_1, ..., p_{n-1}\}$. Hence x^* can be expressed as in the previous chapter equation (1.3)

$$x^* = x_0 + \alpha_0 p_0 + ... + \alpha_{n-1} p_{n-1},$$

where α_i for $i = 0, 1, ..., n-1$ are constants.

Now premultiply both sides of the above equation by $p_k^T A, \quad 0 \leq k \leq n$

$$\begin{aligned} p_k^T A(x^* - x_0) &= \alpha_i p_k^T A p_i \\ \alpha_i &= \frac{p_k^T A(x^* - x_0)}{p_k^T A p_i} \end{aligned} \tag{2.5}$$

Now equaton (2.4)

$$\alpha_k = \frac{p_k^T (b - Ax_0)}{p_k^T A p_k}$$

Therefore,

$$\alpha_k = -\frac{p_k^T r_k}{p_k^T A p_k}. \tag{2.6}$$

Theorem 2.2. *(Conjugate Direction Theorem)*

Let $\{p_0, p_1, ..., p_{n-1}\}$ be a set of nonzero A-orthogonal vectors in $\Re^n$. For any $x_0 \in \Re^n$, consider the sequence $\{x_k\}$ generated according to

$$x_{k+1} = x_k + \alpha_k p_k, \quad k \geq 0 \tag{2.7}$$

with (2.6) and (2.3) converges to the unique solution, x^*, of $Ax = b$ after n steps, that is, $x_k = x^*$.

Proof. Consider $x^* - x_0 \in \Re^n$. Because the p_i's are linearly independent, there exist constants $\beta_i, \quad i = 0, ..., n-1$, such that

$$x^* - x_0 = \beta_0 p_0 + ... + \beta_{n-1} p_{n-1}.$$

Using the equation(1.3) update rules, we have

$$\begin{aligned} x_1 &= x_0 + \beta_0 p_0 \\ x_2 &= x_0 + \beta_0 p_0 + \beta_1 p_1 \\ x_k &= x_0 + \beta_0 p_0 + \beta_1 p_1 + ... + \beta_{k-1} p_{k-1} \\ x_n &= x_0 + \beta_0 p_0 + \beta_1 p_1 + ... + \beta_{n-1} p_{n-1} = x^* \end{aligned} \tag{2.8}$$

We already know

$$\begin{aligned} x^* - x_0 =& \beta_0 p_0 + \beta_1 p_1 + ... + \beta_{n-1} p_{n-1} \\ x_k - x_0 =& \beta_0 p_0 + \beta_1 + ... + \beta_{k-1} p_{k-1}. \end{aligned}$$

Now premultiply both sides of the above equation by $p_k^T A, \quad 0 \leq k \leq n$

$$\begin{aligned} p_k^T A(x^* - x_0) &= p_k^T A(\beta_0 p_0 + ... + \beta_{n-1} p_{n-1}) \\ &= \beta_k p_k^T A p_k \quad for \quad k = 0, 1, ..., n-1, \end{aligned}$$

where the terms $p_k^T A p_i = 0, \quad k \neq i$, by the A-conjugate property equation (2.2).
Hence,

$$\beta_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k}. \tag{2.9}$$

Now, we can write

$$x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + ... + \alpha_{k-1} p_{k-1}.$$

Therefore,

$$x_k - x_0 = \alpha_0 p_0 + \alpha_1 p_1 + ... + \alpha_{k-1} p_{k-1}$$

and premultiplying the above by $p_k^T A$, we obtain

$$\begin{aligned} p_k^T A(x^* - x_0) &= p_k^T A(x^* - x_k + x_k - x_0) \\ &= p_k^T A(x^* - x_k) + 0 \\ &= p_k^T (Ax^* - Ax_k) \\ &= p_k^T (b - Ax_k) \\ &= -p_k^T r_k \end{aligned}$$

because $r_k = Ax^k - b$ and $Ax^* = b$.
Thus,

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

and $x^* = x_n$, which completes the proof. □

Example 2.1. *Find the minimizer of*

$$f(x_1, x_2) = \frac{1}{2} x^T \begin{pmatrix} 4 & 2 \\ 2 & 2 \end{pmatrix} x - x^T \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \ x \in \Re^2.$$

Using the conjugate direction method with the initial point $x_0 = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$, *and A-conjugate directions* $p_0 = \begin{pmatrix} 1 & 0 \end{pmatrix}^T$ *and* $p_1 = \begin{pmatrix} -\frac{3}{8} & \frac{3}{4} \end{pmatrix}^T$.

solution: *We have*

$$r_0 = -b = \begin{pmatrix} 1 & -1 \end{pmatrix}^T,$$

and hence

$$\alpha_0 = -\frac{r_0^T p_0}{p_0^T A p_0} = -\frac{\begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}}{\begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}} = -\frac{1}{4}.$$

Thus,

$$x_1 = x_0 + \alpha_0 p_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \frac{1}{4} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -\frac{1}{4} \\ 0 \end{pmatrix}.$$

To find x_2, *we compute*

$$r_1 = A x_1 - b = \begin{pmatrix} 4 & 2 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} -\frac{1}{4} \\ 0 \end{pmatrix} - \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -\frac{3}{2} \end{pmatrix}$$

and

$$\alpha_1 = -\frac{r_1^T p_1}{p_1^T A p_1} = -\frac{\begin{pmatrix} 0 & -\frac{3}{2} \end{pmatrix} \begin{pmatrix} -\frac{3}{8} \\ \frac{3}{4} \end{pmatrix}}{\begin{pmatrix} \frac{-3}{8} & \frac{3}{4} \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} -\frac{3}{8} \\ \frac{3}{4} \end{pmatrix}} = 2.$$

Therefore,

$$x_2 = x_1 + \alpha_1 p_1 = \begin{pmatrix} -\frac{1}{4} \\ 0 \end{pmatrix} + 2 \begin{pmatrix} -\frac{3}{8} \\ \frac{3}{4} \end{pmatrix} = \begin{pmatrix} -1 \\ \frac{3}{2} \end{pmatrix}.$$

Because f *is a quadratic function in two variables,* $x_2 = x^*$*[2].*

2.1.1 Descent Properties of the Conjugate Direction Method

Let A be a symmetric and positive definite matrix. We define β_k as the subspace of $\Re^n$ spanned by a set of A-orthogonal vectors $\{p_0, p_1, ..., p_{k-1}\}$,

$$\beta_k = span\{p_0, p_1, ..., p_{k-1}\}.$$

Theorem 2.3. *(Expanding Subspace Minimization).*
Let $x_0 \in \Re^n$ be any starting point and suppose that the sequence $\{x_k\}$ is generated by the conjugate direction algorithm (2.6), (2.7). Then

$$r_k^T p_i = 0, \quad for \quad i = 0, 1, ..., k-1, \tag{2.10}$$

and x_k is the minimizer of $f(x) = \frac{1}{2}x^T Ax - b^T x$ over the set

$$\{x | x = x_0 + span\{p_0, p_1, ..., p_{k-1}\}\}. \tag{2.11}$$

Proof. We begin by showing that a point $\hat{x}$ minimizes f over the set (2.10) if and only if $r(\hat{x})^T p_i = 0, \quad for\ each\ i = 0, 1, ..., k-1$. Let us define $h(\sigma) = f(x_0 + \sigma_0 p_0 + ... + \sigma_{k-1} p_{k-1})$, where $\sigma = (\sigma_0, \sigma_1, ..., \sigma_{k-1})^T$. Since $h(\sigma)$ is a strictly convex quadratic, it has a unique minimizer σ^* that satisfies

$$\frac{\partial h(\sigma^*)}{\partial \sigma_i} = 0, \quad i = 0, 1, ..., k-1.$$

By the chain rule, this equation implies that

$$\nabla f(x_0 + \sigma_0^* p_0 + ... + \sigma_{k-1} p_{k-1})^T p_i = 0, \quad i = 0, 1, ..., k-1.$$

By recalling the definition (2.3), we have for the minimizer $\hat{x} = x_0 + \sigma_0^* p_0 + \sigma_1^* p_1 + ... + \sigma_{k-1}^* p_{k-1}$ on the set (2.11) that $r(\hat{x})^T p_i = 0$, as claimed. We now use induction to show that x_k satisfies (2.10). For the case $k = 1$, we have from the previous chapter equation (1.3) $x_1 = x_0 + \alpha_0 p_0$ minimizes f along p_0 that $r_1^T p_0 = 0$. Let us now make the induction hypothesis, namely, that $r_{k-1}^T p_i = 0 \quad for\ i = 0, 1, ..., k-2$. Now

$$r_{k+1} = r_k + \alpha_k A p_k, \tag{2.12}$$

we have

$$r_k = r_{k-1} + \alpha_{k-1} A p_{k-1},$$

so that

$$p_{k-1}^T r_k = p_{k-1}^T r_{k-1} + \alpha_{k-1} p_{k-1}^T A p_{k-1} = 0,$$

by the definition equation(2.6) of α_{k-1}. Meanwhile, for the other vectors $p_i,\ i = 0, 1, ..., k-2$, we have

$$p_i^T r_k = p_i^T r_{k-1} + \alpha_{k-1} p_i^T A p_{k-1} = 0,$$

where $p_i^T r_{k-1} = 0$ because of the induction hypothesis and $p_i^T A p_{k-1} = 0$ because of conjugacy of the vectors p_i. We have shown that $r_k^T p_i = 0, \quad for\ i = 0, 1, ..., k-1$, so the proof is complete. □

2.2 Conjugate Gradient Method

The conjugate gradient method is the conjugate direction method that is obtained by selecting the successive direction vectors as a conjugate version of the successive gradients obtained as the method progresses. Thus, the directions are not specified beforehand, but rather are determined sequentially at each step of the iteration. At step k one evaluates the current gradient vector and adds to it a linear combination of the previous direction vectors to obtain a new conjugate direction vector along which to move[8].

There are three primary advantages to this method of direction selection.
First, unless the solution is attained in less than n steps, the gradient is always nonzero and linearly independent of all previous direction vectors. Indeed, the gradient r_k is orthogonal to the subspace β_k generated by $\{p_0, p_1, ..., p_{k-1}\}$. If the solution is reached before n steps are taken, the gradient vanishes and the process terminates it being unnecessary, in this case, to find additional directions.

Second, a more important advantage of the conjugate gradient method is the especially simple formula that is used to determine the new direction vector. This simplicity makes the method only slightly more complicated than steepest descent.

Third, because the directions are based on the gradients, the process makes good uniform progress toward the solution at every step. This is in contrast to the situation for arbitrary sequences of conjugate directions in which progress may be slight until the final few steps. Although for the pure quadratic problem uniform progress is of no great importance, it is important for generalizations to nonquadratic problems.

2.3 Linear Conjugate Gradient Methods

The conjugate gradient method is an iterative method for solving a linear system of equations[1]

$$Ax = b \tag{2.13}$$

where $A \in \Re^{n \times n}$ matrix that is symmetric and positive definite and $(b, x) \in \Re^n$ vectors.

The problem of a linear system can be stated equivalently as the following quadratic minimization problems: i.e.,

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c, \tag{2.14}$$

that is, both (2.13) and (2.14) have the same unique solution. This equivalence will allow us to interpret the conjugate gradient method either as an algorithm for solving linear systems or as a technique for minimization of convex quadratic functions. The gradient of f equals the residual of the linear system,

$$\nabla f(x) = Ax - b = r(x), \tag{2.15}$$

so in particular at $x = x_k$, we have

$$r_k = Ax_k - b.$$

2.4 Basic Properties of the Conjugate Gradient Method

The conjugate gradient method is a conjugate direction method with a special property. In generating its set of conjugate vectors, it can compute a new vector p_{k+1} by using only the previous vector p_k. It does not need to know all the previous elements $p_0, p_1, ..., p_{k-1}$ of the conjugate set, p_{k+1} is automatically conjugate to these vectors. This remarkable property implies that the method requires little storage and computation[1].

Now for the details of the conjugate gradient method. Each direction p_{k+1} is chosen to be a linear combination of the steepest descent direction $-\nabla f(x_{k+1})$ (which is the same as the residual r_{k+1}, by (2.3)) and the previous direction p_k. We write

$$p_{k+1} = -r_{k+1} + \beta_k p_k, \tag{2.16}$$

where the scalar β_k is to be determined by the requirement that p_k and p_{k+1} must be conjugate with respect to A. By premultiplying equation (2.16) by $p_k^T A$ and imposing the condition $p_k^T A p_{k+1} = 0$, we find that

$$\begin{aligned} p_k^T A p_{k+1} = -p_k^T A r_{k+1} + \beta_k p_k^T A p_k &= 0 \\ p_k^T A r_{k+1} &= \beta_k p_k^T A p_k \\ \beta_k &= \frac{r_{k+1}^T A p_k}{p_k^T A p_k}. \end{aligned} \tag{2.17}$$

We choose the first search direction p_0 to be the steepest descent direction at the initial point x_0. As in the general conjugate direction method, we perform successive one-dimensional minimizations along each of the search directions. We have thus specified a complete algorithm, which we express formally as follows.

Algorithm 2.1. *(C-G)*
step(1.) *set* $k = 0$; *select the initial point* $x_0 \in \Re^n$, $\epsilon > 0$;
step(2.) $r_0 = Ax_0 - b; p_0 = -r_0$;
while $\|r_k\| \geq \epsilon$
step(3) $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$;
setp(4) $x_{k+1} = x_k + \alpha_k p_k$;
step(5) $r_{k+1} = Ax_{k+1} - b$;
step(7) $\beta_k = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$;
step(8) $p_{k+1} = r_{k+1} + \beta_k p_k$;
step(9) *set* $k = k + 1$;
end (% end while)

This version is useful for studying the essential properties of the conjugate gradient method, but we present a more efficient version later. We show first that the directions $\{p_0, p_1, ..., p_{n-1}\}$ are indeed conjugate, which by Theorem 2.2 implies termination in n steps. The theorem below establishes this property and two other important properties. First, the residuals r_i are mutually orthogonal. Second, each search direction p_k and residual r_k is contained in the Krylov subspace of degree k for r_0, defined as

$$K(r_0, k) = span\{r_0, Ar_0, ..., A^k r_0\}.$$

Theorem 2.4. *(Theorem of conjugate gradient method)*
Suppose that the k^{th} *iterate generated by the conjugate gradient method is not the solution point* x^*. *The following four properties hold:*

$$r_k^T r_i = 0, \quad for \quad i = 0, ..., k-1, \tag{2.18}$$

$$span\{r_0, r_1, ..., r_k\} = span\{r_0, Ar_0, ..., A^k r_0\}, \tag{2.19}$$

$$span\{p_0, p_1, ..., p_k\} = span\{r_0, Ar_0, ..., A^k r_0\}, \tag{2.20}$$

$$p_k^T A p_i = 0, \quad for \quad i = 0, 1, ..., k-1. \tag{2.21}$$

Therefore, the sequence $\{x_k\}$ *converges to* x^* *in at most* n *steps.*

Proof. **proof by induction:** The expressions (2.19) and (2.20) holds true for $k = 0$. suppose (2.19) and (2.20) holds true for some k. Now we show that they also holds for $k+1$. Now

$$r_{k+1} = r_k + \alpha_k A p_k \tag{2.22}$$

$$r_k \in span\{r_0, Ar_0, ..., A^k r_0\}, \quad p_k \in span\{r_0, Ar_0, ..., A^k r_0\},$$

premultiplying the second equaton by A, we obtain

$$Ap_k \in span\{Ar_0, ..., A^{k+1} r_0\}. \tag{2.23}$$

applying (2.22), we find that

$$r_{k+1} \in span\{r_0, Ar_0, ..., A^{k+1}r_0\}.$$

combining this expression with the induction hypothesis for (2.19), we conclude that

$$span\{r_0, r_1, ..., r_k, r_{k+1}\} \in span\{r_0, Ar_0, ..., A^{k+1}r_0\}.$$

To prove that the reverse inclusion holds as well, we use the induction hypothesis on (2.18) to deduce that

$$A^{k+1}r_0 = A(A^k r_0) \in span\{Ap_0, Ap_1, ..., Ap_k\}.$$

Since by (2.22) we have $Ap_i = (r_{i+1} - r_i)/\alpha_i$ for $i = 0, 1, ..., k$, it follows that

$$A^{k+1}r_0 \in span\{r_0, r_1, ..., r_{k+1}\}.$$

combining this expression with the induction hypothesis for (2.19), we find that

$$span\{r_0, Ar_0, ..., A^{k+1}r_0\} \subset span\{r_0, r_1, ..., r_k, r_{k+1}\}.$$

Therefore,

$$span\{r_0, r_1, ..., r_k, r_{k+1}\} = span\{r_0, Ar_0, ..., A^{k+1}r_0\},$$

for k replace by $k + 1$.

We show that $span\{p_0, p_1, ..., p_k\} = span\{r_0, Ar_0, ..., A^k r_0\}$, continues to hold when k is replaced by $k + 1$ by the following argument:

$$\begin{aligned}
span\{p_0, p_1, ..., p_k, p_{k+1}\} =& span\{p_0, p_1, ..., p_k, r_{k+1}\}, \quad \textit{by algorithm (2.1) in step (7)} \\
=& span\{r_0, Ar_0, ..., A^k r_0, r_{k+1}\}, \qquad \textit{by induction hypothesis for (2.20)} \\
=& span\{r_0, r_1, ..., r_k, r_{k+1}\}, \qquad \textit{by (2.19)} \\
=& span\{r_0, Ar_0, ..., A^k r_0, A^{k+1}r_0\}, \qquad \textit{by (2.19)} \quad k+1.
\end{aligned}$$

Therefore, $span\{p_0, p_1, ..., p_k, p_{k+1}\} = span\{r_0, Ar_0, ..., A^k r_0, A^{k+1}r_0\}$, for k replaced by $k+1$.

Next, we prove the conjugacy condition (2.21) with k replaced by $k + 1$.

Multiplying algorithm (2.1) in step (7) by $Ap_i, i = 0, 1, ..., k$, we obtain

$$p_{k+1}^T Ap_i = -r_{k+1}^T Ap_i + \beta_k p_k^T Ap_i. \tag{2.24}$$

By the definition (2.17) of β_k, the right-hand-side of (2.24) vanishes when $i = k$. For $i \leq k-1$ we need to collect a number of observations. Note first that our induction hypothesis for (2.20) implies that the directions $p_0, p_1, ..., p_k$ are conjugate, so we can apply Theorem 2.3 to deduce that

$$r_{k+1}^T p_i = 0, \quad for \quad i = 0, 1, ..., k. \tag{2.25}$$

Second, by repeatedly applying (2.20), we find that for $i = 0, 1, ..., k-1$, the following inclusion holds:

$$Ap_i \in Aspan\{r_0, Ar_0, ..., A^i r_0\} = span\{Ar_0, A^2 r_0, ..., A^{i+1} r_0\} \subset span\{p_0, p_1, ..., p_{i+1}\}. \tag{2.26}$$

combining (2.25) and (2.26), we deduce that

$r_{k+1}^T Ap_i = 0, \quad$ for $\; i = 0, 1, ..., k-1,$
so the first term in the right-hand-side of (2.20) vanishes for $i = 0, 1, ..., k-1$.
Because of the induction hypothesis for (2.21), the second term vanishes as well, and we conclude that

$$p_{k+1}^T Ap_i = 0, \quad i = 0, 1, ..., k.$$

Hence, the induction argument holds for (2.21) also.

Finally, we prove (2.20) by a noninductive argument. Because the direction set is conjugate, we have that from Theorem 2.3

$r_k^T p_i = 0$ for all $i = 0, 1, ..., k-1$ and any $k = 1, 2, ..., n-1$. By rearranging algorithm (2.1) in step (7) , we find that

$$p_i = -r_i + \beta_i p_{i-1},$$

so that $r_i \in span\{p_i, p_{i-1}\}$ for all $i = 1, ..., k-1$.

We conclude that

$r_k^T r_i = 0$ for all $i = 1, ..., k-1$.

Therefore, the sequence $\{x_k\}$ converges to x^* in at most n steps. □

2.5 A Practical form of the Conjugate Gradient Method

We can derive a slightly more cheap and time consuming form of the conjugate gradient method by using the results of Theorem 2.3 and Theorem 2.4 First, we can use algorithm 2.1 step(8) and (2.10) to replace the formula algorithm 2.1 step(3) for α_k by formula

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}, \quad for \quad \alpha_k.$$

By applying Theorem 2.3 $r_k^T p_i = 0$. Also, since $p_k = -r_k + \beta_{k-1} p_{k-1}$, and premultiplying this by r_k^T, we get

$$r_k^T p_k = -r_k^T r_k + \beta_{k-1} r_k^T p_{k-1} = -r_k^T r_k$$

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

Second, we have from (2.7) that $\alpha_k A p_k = r_{k+1} - r_k$,

so by applying $p_k = -r_k + \beta_{k-1} p_{k-1}$ and $r_k^T p_i = 0, \quad for\ all\ i = 0, 1, ..., k-1$ once. again we can simplify the formula for β_k to replace the formula

$$\beta_k = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}.$$

Substituting this into the original equation for β_k gives

$$\beta_k = \frac{r_{k+1}^T [r_{k+1} - r_k]}{p_k^T [r_{k+1} - r_k]},$$

by applying Theorem 2.3 $r_k^T p_i = 0$. Also, since $p_k = -r_k + \beta_{k-1} p_{k-1}$, and premultiplying this by r_k^T, we get

$$r_k^T p_k = -r_k^T r_k + \beta_{k-1} r_k^T p_{k-1} = -r_k^T r_k,$$

where once again we used Theorem 2.3. Hence, we get

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

By using these formula together with (2.7), we obtain the following standard form of the conjugate gradient method.

Algorithm 2.2. *C-G*

step(1) *set* $k = 0$; select the initial point $x_0 \in \Re^n$, $\epsilon > 0$;
step(2) $r_0 = Ax_0 - b, \qquad p_0 = -r_0,$
while $\|r_k\| \geq \epsilon$
step(3) $\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k};$
step(4) $x_{k+1} = x_k + \alpha_k p_k;$
step(5) $r_{k+1} = r_k + \alpha_k A p_k;$
step(6) $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k};$
step(7) $p_{k+1} = r_{k+1} + \beta_k p_k;$
step(8) *set* $k = k + 1$; go to step 3
end(%end while)

At any given point in Algorithm 2.2 we never need to know the vectors x, r, and p for more than the last two iterations. Accordingly, implementations of this algorithm over write old values of these vectors to save on storage. The major computational tasks to be performed at each step are computation of the matrix-vector product Ap_k, calculation of the inner products $p_k^T(Ap_k)$ and $r_{k+1}^T r_{k+1}$, and calculation of three vector sums. The inner product and vector sum operations can be performed in a small multiple of n floating-point operations, while the cost of the matrix-vector product is, of course, dependent on the problem. The CG method is recommended only for large problems, otherwise, Gaussian elimination or other factorization algorithms such as the singular value decomposition are to be preferred, since they are less sensitive to rounding errors. For large problems, the CG method has the advantage that it does not alter the coefficient matrix and (in contrast to factorization techniques) does not produce fill in the arrays holding the matrix. Another key property is that the CG method sometimes approaches the solution quickly, as we discuss next.

2.6 The Convergence Rate of Conjugate Gradient Method

We have seen that in exact arithmetic the conjugate gradient method will terminate at the solution in at most n iterations. What is more remarkable is that when the distribution of the eigenvalues of A has certain favorable features, the algorithm will identify the solution in many fewer than n iterations. To explain this property, we begin by viewing the expanding subspace minimization property proved in Theorem 2.3 in a slightly different way, using it to show that Algorithm 2.3 is optimal in a certain important sense.

From algorithm 2.2 step(4) and (2.20), we have that

$$x_{k+1} = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + ... + \alpha_k p_k \tag{2.27}$$

$$= x_0 + \gamma_0 r_0 + \gamma_1 A r_0 + ... + \gamma_k A^k r_0, \tag{2.28}$$

for some constants γ_i. We now define $P_k^*(.)$ to be a polynomial of degree k with coefficients $\gamma_0, \gamma_1, ..., \gamma_k$. Like any polynomial, P_k^* can take either a scalar or a square matrix as its argument. For the matrix argument A, we have

$$P_k^*(A) = \gamma_0 I + \gamma_1 A + ... + \gamma_k A_k,$$

which allows us to express (2.28) as follows:

$$x_{k+1} = x_0 + P_k^*(A) r_0. \tag{2.29}$$

We will now see that among all possible methods whose first k steps are restricted to the Krylov subspace $K(r_0, k)$ given by (2.28), Algorithm 2.2 does the best job of minimizing the distance to the solution after k steps, when this distance is measured by the weighted norm measure $\|.\|_A$ defined by

$$\|z\|_A^2 = z^T A z. \tag{2.30}$$

Using this norm and the definition of f (2.14), and x^* is a minimizer of f, it is easy to show that

$$\begin{aligned} f(x) - f(x^*) =& \frac{1}{2} x^T A x - b^T x \\ =& \frac{1}{2} \|x - x^*\|_A^2. \end{aligned} \tag{2.31}$$

Theorem 2.4 states that x_{k+1} minimizes f, and hence $\|x - x^*\|_A^2$, over the set $x_0 + span\{p_0, p_1, ..., p_k\}$, which by (2.20) is the same as $x_0 + span\{r_0, Ar_0, ..., A^k r_0\}$. It follows from (2.29) that the polynomial P_k^* solves the following problem in which the minimum is taken over the space of all possible polynomials of degree k:

$$\min_{P_k} \|x_0 + P_k(A) r_0 - x^*\|_A. \tag{2.32}$$

We exploit this optimality property repeatedly in the remainder of the section.Since

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

we have that

$$x_{k+1} - x^* = x_0 + P_k^*(A)r_0 - x^* = [1 + P_k^*(A)A](x_0 - x*). \tag{2.33}$$

Let $0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ be the eigenvalues of A, and let $v_1, v_2, ..., v_n$ be the corresponding orthonormal eigenvectors, so that

$$Av_i = \lambda_i v_i \Rightarrow A = \lambda_i v_i v_i^{-1},$$

by definition of orthonormal matrix

$$A = \lambda_i v_i v_i^T$$

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

Since the eigenvectors span the whole space $\Re^n$, we can write

$$x_0 - x* = \sum_{i=1}^{n} \zeta_i v_i, \tag{2.34}$$

for some coefficients ζ_i . It is easy to show that any eigenvector of A is also an eigenvector of $P_k(A)$ for any polynomial P_k. For our particular matrix A and its eigenvalues λ_i and eigenvectors v_i, we have

$$P_k(A)v_i = P_k(\lambda_i)vi, i = 1, 2, ..., n.$$

By substituting (2.34) into (2.33) we have

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

By using the fact that $\|z\|_A^2 = z^T Az = \sum_{i=1}^n \lambda_i (v_i^T z)^2$, we have

$$\|x_{k+1} - x^*\|_A^2 = \sum_{i=1}^{n} \lambda_i [1 + \lambda_i P_k(\lambda_i)]^2 \zeta_i^2. \tag{2.35}$$

Since the polynomial P_k^* generated by the CG method is optimal with respect to this norm,we have

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

By extracting the largest of the terms $[1 + \lambda_i P_k(\lambda_i)]^2$ from this expression, we obtain that

$$\begin{aligned}\|x_{k+1} - x^*\|_A^2 &\leq \min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 (\sum_{j=1}^{n} \lambda_j \zeta_j^2 \\ &= \min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 \|x_0 x^*\|_A^2, \end{aligned} \tag{2.36}$$

where we have used the fact that

$$\|x_0 - x^*\|_A^2 = \sum_{j=1}^{n} \lambda_j \zeta_j^2.$$

The expression (2.36) allows us to quantify the convergence rate of the CG method by estimating the nonnegative scalar quantity

$$\min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2. \tag{2.37}$$

In other words, we search for a polynomial P_k that makes this expression as small as possible. In some practical cases, we can find this polynomial explicitly and draw some interesting conclusions about the properties of the CG method. The following result is an example.

Theorem 2.5. ***(Supper linear convergence)***
If A has only r distinct eigenvalues, then the CG iteration will terminate at the solution in at most r iterations.

Proof. Suppose that the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ take on the r distinct values $\tau_1 < \tau_2 < ... < \tau_r$. We define a polynomial $Q_r(\lambda)$ by

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

and note that $Q_r(\lambda_i) = 0$ for $i = 1, 2, ..., n$ and $Q_r(0) = 1$. From the latter observation, we deduce that $Q_r(\lambda) - 1$ is a polynomial of degree r with a root at $\lambda = 0$, so by polynomial division, the function $\bar{P}_{r-1}$ defined by

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

is a polynomial of degree $r - 1$. By setting $k = r - 1$ in (2.37), we have

$$\begin{aligned} 0 &\leq \min_{P_{r-1}} \max_{1 \leq i \leq n} [1 + \lambda_i P_{r-1}(\lambda_i)]^2 \\ &\leq \max_{1 \leq i \leq n} [1 + \lambda_i \bar{P}_{r-1}(\lambda_i)]^2 \\ &= \max_{1 \leq i \leq n} [1 + \lambda_i \frac{(Q_r(\lambda_i) - 1)}{\lambda_i}]^2 \\ &= \max_{1 \leq i \leq n} Q_r^2(\lambda_i) = 0. \end{aligned}$$

Hence, the constant in (2.37) is zero for the value $k = r - 1$, so we have by substituting into (2.36) that $\|x_r - x^*\|_A^2 = 0$.
Therefore $x_r = x_*$, as claimed. □

Chapter 3

Nonlinear Conjugate Gradient Methods

They are extensions of the conjugate gradient iterative method for solving linear systems adapted to solve unconstrained nonlinear optimization problems[1].

$$\min_{x \in \Re^n} f(x), \tag{3.1}$$

where $f : \Re^n \to \Re$ is continuously differentiable. We have noted that the CG method, Algorithm 2.2, can be viewed as a minimization algorithm for the convex quadratic function f defined by (2.14). It is natural to ask whether we can adapt the approach to minimize general convex functions, or even general nonlinear functions f . In fact, as we show in this chapter, nonlinear variants of the conjugate gradient are well studied and have proved to be quite successful in practice.

3.1 The Fletcher-Reeves Method

Fletcher and Reeves [1] showed how to extend the conjugate gradient method to nonlinear functions by making two simple changes in Algorithm 2.2. First, in place of the formula $\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$ for the step length α_k (which minimizes f along the search direction p_k), we need to perform a line search that identifies an approximate minimum of the nonlinear function f along p_k. Second, the residual r, which is simply the gradient of f in Algorithm 2.2, must be replaced by the gradient of the nonlinear objective f. These changes give rise to the following algorithm for nonlinear optimization.

Algorithm 3.1. *(FR-CG)*
step(1.) *Given* x_0, $k = 0$, $\epsilon > 0$;
step(2.) *Evaluate* $f_0 = f(x_0)$, $\nabla f_0 = \nabla f(x_0)$;
step(3.) *Set* $p_0 = -\nabla f_0$;
while $\|\nabla f_k\| \geq \epsilon$

step(4.) *compute* α_k *in a minimizer of one dimensional line search:*

$$\min_{\alpha>0} f(x_k + \alpha p_k)$$

step(5.) *set* $x_{k+1} = x_k + \alpha_k p_k$*;*
step(6.) *Evaluate* $\nabla f_{k+1} = \nabla f(x_{k+1})$*;*
step(7.) $\beta_k^{FR} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$*;*
step(8.) $p_{k+1} = -\nabla f_{k+1} + \beta_k^{FR} p_k$*;*
step(9.) $k = k + 1$*;*
end(%end while)

If we choose f to be a strictly convex quadratic and α_k to be the exact minimizer, this algorithm reduces to the linear conjugate gradient method, Algorithm 2.2. Algorithm 3.1 is appealing for large nonlinear optimization problems because each iteration requires only evaluation of the objective function and its gradient. No matrix operations are required for the step computation, and just a few vectors of storage are required. To make the specification of Algorithm 3.1 complete, we need to be more precise about the choice of line search parameter α_k. Because of algorithm 3.1 in step (8), the search direction p_k may fail to be a descent direction unless α_k satisfies certain conditions.

By taking the inner product of algorithm 3.1 in step (8) (with k replacing $k+1$) with the gradient vector ∇f_k, we obtain

$$\nabla f_k^T p_k = -\|\nabla f_k\|^2 + \beta_{k-1}^{FR} \nabla f_k^T p_{k-1}. \tag{3.2}$$

If the line search is exact, so that α_{k-1} is a local minimizer of f along the direction p_{k-1}, we have that $\nabla f_k^T p_{k-1} = 0$. In this case we have from (3.2) that $\nabla f_k^T p_k < 0$, so that p_k is indeed a descent direction. If the line search is not exact, however, the second term in (3.2) may dominate the first term, and we may have $\nabla f_k^T p_k > 0$, implying that p_k is actually a direction of ascent. We can avoid this situation by requiring the step length α_k to satisfy the strong Wolfe conditions, which we restate here:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \tag{3.3}$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq -c_2 \nabla f_k^T p_k, \tag{3.4}$$

where $0 < c_1 < c_2 < \frac{1}{2}$. (Note that we impose $c_2 < \frac{1}{2}$ here, in place of the looser condition $c_2 < 1$ that was used in the chapter 1). By applying Lemma 3.1 below, we can show that condition (3.4) implies that (3.2) is negative, and we conclude that any line search procedure that yields an α_k satisfying (3.3) and (3.4) will ensure that all directions p_k are descent directions for the function f.

3.2 The Polak-Ribiere Method and Variants

There are many variants of the Fletcher-Reeves method that differ from each other mainly in the choice of the parameter β_k. An important variant, proposed by Polak and Ribiere[1],

defines this parameter as follows :

$$\beta_k^{PR} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}. \tag{3.5}$$

We refer to the Algorithm in which equation (3.5) replaces to Algorithm (3.1) in step (7) as Algorithm PR. It is identical to Algorithm FR when f is a strictly convex quadratic function and the line search is exact, since by (2.18) the gradients are mutually orthogonal, and so

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

Therefore,$\beta_k^{PR} = \beta_k^{FR}$.

The PR Algorithm is that the strong Wolfe conditions do not guarantee that p_k is always a descent direction. If we define the β_k parameter as

$$\beta_k^{PR} = \max\{\beta_k^{PR}, 0\}, \tag{3.6}$$

giving rise to an algorithm we call Algorithm PR, then a simple adaptation of the strong Wolfe conditions ensures that the descent property holds.

There are many other choices for β_k that coincide with the Fletcher-Reeves formula β_k^{FR} in the case where the objective is quadratic and the line search is exact.

$$\beta_k^{HS} = \frac{\nabla f_k^T(\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k)^T p_k}, \tag{3.7}$$

is called HS Algorithm that is similar to Algorithm PR, both in terms of its theoretical convergence properties and in its practical performance. Formula (3.7) can be derived by demanding that consecutive search directions be conjugate with respect to the average Hessian over the line segment $[x_k, x_{k+1}]$, which is defined as

$$G_k \equiv \int_0^1 [\nabla^2 f(x_k + t\alpha_k p_k)]dt.$$

Recalling from Taylors theorem (Theorem 1.2) that $\nabla f_{k+1} = \nabla f_k + \alpha_k G_k p_k$, we see that for any direction of the form $p_{k+1} = -\nabla f_{k+1} + \beta_k p_k$, the condition $p_{k+1}^T G_k p_k = 0$ requires β_k to be given by (3.7). Later, we see that it is possible to guarantee global convergence for any parameter β_k satisfying the bound

$$|\beta_k| \leq \beta_k^{FR}, \tag{3.8}$$

for all $k \geq 2$. This fact suggests the following modification of the PR method, which has performed well on some applications. For all $k \geq 2$ let

$$\beta_k = \begin{cases} -\beta_k^{FR}, & \text{if } \beta_k^{PR} < -\beta_k^{FR} \\ \beta_k^{PR}, & \text{if } |\beta_k^{PR}| \leq \beta_k^{FR} \\ \beta_k^{FR}, & \text{if } \beta_k^{PR} > \beta_k^{FR}. \end{cases} \tag{3.9}$$

The algorithm based on this strategy will be denoted by FR-PR. Other variants of the CG method have recently been proposed. Two choices for β_k that possess attractive theoretical and computational properties are

$$\beta_k = \frac{\|\nabla f_{k+1}\|^2}{(\nabla f_{k+1} - \nabla f_k)^T p_k} \tag{3.10}$$

and

$$\beta_k = (\hat{y}_k - 2p_k \frac{\|\hat{y}_k\|^2}{\hat{y}_k^T p_k})^T \frac{\nabla f_{k+1}}{\hat{y}_k^T p_k}, \quad \textit{with} \quad \hat{y}_k = \nabla f_{k+1} - \nabla f_k. \tag{3.11}$$

These two choices guarantee that p_k is a descent direction, provided the step length α_k satisfies the Wolfe conditions. The CG algorithms based on (3.10) or (3.11) appear to be competitive with the PolakRibiere method.

3.3 Behavior of the Fletcher-Reeves Method

We now investigate the Fletcher-Reeves algorithm, Algorithm 3.1, a little more closely, proving that it is globally convergent and explaining some of its observed inefficiencies. The following result gives conditions on the line search under which all search directions are descent directions. It assumes that the level set $L = \{x : f(x) \leq f(x_0)\}$ is bounded and that f is twice continuously differentiable, so that we have from Lemma 1.1 that there exists a step length α_k satisfying the strong Wolfe condition [1].

Lemma 3.1. *Suppose that Algorithm 3.1 is implemented with a step length α_k that satisfies the strong Wolfe conditions (3.3) and (3.4) with $0 < c_2 < \frac{1}{2}$. Then the method generates descent directions p_k that satisfy the following inequalities:*

$$-\frac{1}{1-c_2} \leq \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2} \leq \frac{2c_2 - 1}{1 - c_2}, \quad \textit{for all } k = 0, 1, ... \tag{3.12}$$

Proof. Define the function $t(\delta) = \frac{(2\delta - 1)}{(1-\delta)}$ is monotonically increasing on the interval $[0, \frac{1}{2}]$ and that $t(0) = -1$ and $t(\frac{1}{2}) = 0$. Hence, because of $c_2 \in (0, \frac{1}{2})$, we have

$$-1 < \frac{2c_2 - 1}{1 - c_2} < 0. \tag{3.13}$$

The descent condition $\nabla f_k^T p_k < 0$ follows immediately once we establish (3.12). Proof by induction: For $k = 0$, the middle term in (3.12) is -1, so by using (3.13), we see that both inequalities in (3.12) are satised. Next, assume that (3.12) holds for some $k \geq 1$. From algorithm 3.1 in step (7) and step (8), we have

$$\beta_k = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$$

and

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

premultiplying the second equation by ∇f_{k+1}

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

$$\frac{\nabla f_{k+1}^T p_{k+1}}{\|\nabla f_{k+1}\|^2} = -1 + \beta_k \frac{\nabla f_{k+1}^T pk}{\|\nabla f_{k+1}\|^2} = -1 + \frac{\nabla f_{k+1}^T pk}{\|\nabla f_k\|_2}. \tag{3.14}$$

By using the line search condition (3.4), we have

$$|\nabla f_{k+1}^T p_k| \leq -c_2 \nabla f_k^T p_k,$$

so by combining with (3.14) and recalling β_k^{FR} formula, we obtain

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

Substituting for the term $\frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2}$ from the left-hand-side of the induction hypothesis (3.14), we obtain

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

which shows that (3.14) holds for $k + 1$ as well. □

This result used only the second strong Wolfe condition (3.4), the first Wolfe condition (3.3) will be needed in the next section to establish global convergence. The bounds on $\nabla f_k^T p_k$ in (3.12) impose a limit on how fast the norms of the steps $\|p_k\|$ can grow, and they will play a crucial role in the convergence analysis given below.

Lemma 3.1 can also be used to explain a weakness of the Fletcher-Reeves method. We will argue that if the method generates a bad direction and a tiny step, then the next direction and next step are also likely to be poor. Let θ_k denote the angle between p_k and the steepest descent direction $-\nabla f_k$, dened by

$$\cos \theta_k = - \frac{\nabla f_k^T p_k}{\|\nabla f_k\| \|p_k\|}, \tag{3.15}$$

Suppose that p_k is a poor search direction, in the sense that it makes an angle of nearly 90^0 with $-\nabla f_k$, that is, $\cos\theta_k \approx 0$. By multiplying both sides of (3.12) by $\frac{\|\nabla f_k\|}{\|p_k\|}$ and using (3.14), we obtain

$$\frac{1-2c_2}{1-c_2}\frac{\|\nabla f_k\|}{\|p_k\|} \leq -\frac{\nabla f_k^T p_k}{\|\nabla f_k\|\|pk\|} = \cos\theta_k \leq \quad \frac{1}{1-c_2}\frac{\|\nabla f_k\|}{\|p_k\|}, \quad for\ all \quad k = 0, 1, ... \quad (3.16)$$

From these inequalities, we deduce that $\cos\theta_k \approx 0$ if and only if

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

Since p_k is almost orthogonal to the gradient, it is likely that the step from x_k to x_{k+1} is tiny, that is, $x_{k+1} \approx x_k$. If so, we have $\nabla f_{k+1} \approx \nabla f_k$, and therefore

$$\beta_k^{FR} \approx 1, \quad (3.17)$$

by the definition algorithm 3.1 in step (7). By using this approximation together with $\|\nabla f_{k+1}\| \approx \|\nabla f_k\| \ll \|p_k\|$ in algorithm 3.1 step (8), we conclude that

$$p_{k+1} \approx p_k,$$

so the new search direction will improve little (if at all) on the previous one. It follows that if the condition $\cos\theta_k \approx 0$ holds at some iteration k and if the subsequent step is small, a long sequence of unproductive iterates will follow.

The Polak-Ribiere method behaves quite differently in these circumstances. If, as in the previous paragraph, the search direction p_k satisfies $\cos\theta_k \approx 0$ for some k, and if the subsequent step is small, it follows by substituting $\nabla f_k \approx \nabla f_{k+1}$ into (3.5) that $\beta_k^{PR} \approx 0$. From the formula step (8), we find that the new search direction p_{k+1} will be close to the steepest descent direction $-\nabla f_{k+1}$, and $\cos\theta_{k+1}$ will be close to 1. Therefore, Algorithm PR essentially performs a restart after it encounters a bad direction. The same argument can be applied to Algorithms (PR)and HS. For the FR-PR variant, defined by (3.9), we have noted already that $\beta_k^{FR} \approx 1$, and $\beta_k^{PR} \approx 0$. The formula (3.9) thus sets $\beta_k = \beta_k^{PR}$, as desired. Thus, the modification (3.9) seems to avoid the inefficiencies of the FR method, while falling back on this method for global convergence.

3.4 Global Convergence

Unlike the linear conjugate gradient method, whose convergence properties are well understood and which is known to be optimal as described above, nonlinear conjugate gradient methods possess surprising, sometimes bizarre, convergence properties. We now present a few of the main results known for the Fletcher-Reeves and Polak-Ribiere methods using practical line searches[1].

For the purposes of this section, we make the following (nonrestrictive) assumptions on the objective function[1].

Assumption 3.1.

i) The level set $L := \{x : f(x) \leq f(x_0)\}$ *is bounded.*

ii) In some neighborhood $\aleph$ *of* L*, the objective function* f *is Lipschizt continuously differentiable, that is there exists a constant* $L > 0$ *such that*

$$\|r(x) - r(\hat{x})\| \leq L\|x - \hat{x}\|, \quad for\ all\ \ x, \hat{x} \in \aleph. \tag{3.18}$$

These assumptions imply that there is a constant γ *such that*

$$\|\nabla f(x)\| \leq \bar{\gamma}, \quad for\ \ all\ \ x \in L \tag{3.19}$$

Our main analytical tool in this section is Zoutendijks theorem 1.3. It states, that under Assumptions 3.1, any line search iteration of the form $x_{k+1} = x_k + \alpha_k p_k$ *, where* p_k *is a descent direction and* α_k *satisfies the strong Wolfe conditions(3.3) and (3.4) gives the limit*

$$\sum_{k=0}^{\infty} \cos^2 \theta_k \|\nabla f_k\|^2 < \infty. \tag{3.20}$$

We can use this result to prove global convergence for algorithms that are periodically restarted by setting $\beta_k = 0$*. If* k_1, k_2*, and so on denote the iterations on which restarts occur, we have from (3.20) that*

$$\sum_{k_1, k_2, \ldots} \|\nabla f_k\|^2 < \infty. \tag{3.21}$$

If we allow no more than n *iterations between restarts, the sequence* $\{k_j\}_{j=1}^{\infty}$ *is infinite, and from (3.21) we have that* $\lim_{j\to\infty} \|\nabla f_{k_j}\| = 0$*. That is, a subsequence of gradients approaches zero, or equivalently,*

$$\liminf_{k\to\infty} \|\nabla f_k\| = 0. \tag{3.22}$$

We can build on Lemma 3.1 and Zoutendijks result (3.21) to prove a global convergence result for the Fletcher-Reeves method. While we cannot show that the limit of the sequence of gradients $\{\nabla f_k\}$ *is zero, the following result shows that this sequence is not bounded away from zero.*

Theorem 3.1. *Suppose that Assumptions 3.1 hold, and that Algorithm 3.1 is implemented with a line search that satisfies the strong Wolfe conditions (3.3) and (3.4), with* $0 < c_1 < c_2 < 1$*. Then*

$$\lim inf_{k\to\infty} \|\nabla f_k\| = 0. \tag{3.23}$$

Proof. Proof by contradiction: Assume that the opposite of (3.23) holds, there is a constant $\gamma > 0$ such that

$$\|\nabla f_k\| \geq \gamma, \tag{3.24}$$

for all k sufficiently large. By substituting the left inequality of (3.16) into Zoutendijks condition (3.20), we obtain

$$\sum_{k=0}^{\infty} \frac{\|\nabla f_k\|^2}{\|p_k\|^2} \|\nabla f_k\|^2 = \sum_{k=0}^{\infty} \frac{\|f_k\|^4}{\|p_k\|^2} < \infty. \tag{3.25}$$

By using (3.4) and (3.12), we obtain that

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

since

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

$$|\nabla f_k^T p_{k-1}| \leq -c_2 \nabla f_{k-1}^T p_{k-1} \leq \frac{c_2}{1-c_2} \|\nabla f_{k-1}\|^2. \tag{3.26}$$

Thus, from (step 8) and recalling the definition (step 7) of β_k^{FR}, we obtain $p_k = -\nabla f_k + \beta_{k-1} p_{k-1}$, squaring both sides

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

Applying this relation repeatedly, and define $c_3 = \frac{(1+c_2)}{(1-c_2)} \geq 1$, we have

$$\begin{aligned}
\|p_k\|^2 \leq & c_3 \|\nabla f_k\|^2 + (\beta_{k-1}^{FR})^2 (c_3 \|\nabla f_{k-1}\|^2 + (\beta_{k-2}^{FR})^2 (c_3 \|\nabla f_{k-3}\|^2 + \ + (\beta_0^{FR})^2 \|p_0\|^2)) ...) \\
= & c_3 \|\nabla f_k\|^2 + (\frac{\nabla f_k^T \nabla f_k}{\nabla f_{k-1}^T \nabla f_{k-1}})^2 (c_3 \|\nabla f_{k-1}\|^2 + (\frac{\nabla f_{k-1}^T \nabla f_{k-1}}{\nabla f_{k-2}^T \nabla f_{k-2}^T})^2 (c_3 \|\nabla f_{k-3}\|^2 \\
& + ... + (\frac{\nabla f_1^T \nabla f_1}{\nabla f_0^T \nabla f_0})^2 \|p_0\|^2)) ...)
\end{aligned}$$

$$= c_3 \|\nabla f_k\|^4 \sum_{j=0}^{k} \|\nabla f_j\|^{-2}, \tag{3.27}$$

where

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

and $p_0 = -\nabla f_0$. By using the bounds (3.19) and (3.24) in (3.27), we obtain

$$\|p_k\|^2 \leq \frac{c_3\bar{\gamma}^4}{\gamma^2}k \tag{3.28}$$

which implies that

$$\sum_{k=1}^{\infty}\frac{1}{\|p_k\|^2} \geq \gamma^4\sum_{k=1}^{\infty}\frac{1}{k} \tag{3.29}$$

for some positive constant γ^4. On the other hand, from (3.24) and (3.25), we have that

$$\sum_{k=1}^{\infty}\frac{1}{\|p_k\|^2} < \infty. \tag{3.30}$$

However, if we combine this inequality with (3.29), we obtain that $\sum_{k=1}^{\infty}\frac{1}{k} < \infty$, which is not true. Hence, (3.24) does not hold, and the claim (3.23) is proved.

This global convergence result can be extended to any choice of β_k satisfying (3.8), and in particular to the FR-PR method given by (3.9).

In general, if we can show that there exist constants $c_4, c_5 > 0$ such that

$$\cos\theta_k \geq c_4\|\nabla f_k\|\|p_k\|, \quad \|\nabla f_k\|\|p_k\| \geq c_5 > 0, k = 1, 2, ...,$$

it follows from (3.20) that

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

In general, this result can be established for the Polak-Ribiere method under the assumption that f is strictly convex and that an exact line search is used. □

Chapter 4

Numerical Implementation

We use the following test problems to compare the performance of the Conjugate gradient method and its Extension with the steepest descent and Newtons methods.

Test Problems

$$
\begin{aligned}
T1 \quad f(x,y,z) =& \frac{3}{2}x^2 + 2y^2 + \frac{3}{2}z^2 + xz + 2yz - 3x - z \\
T2 \quad f(x,y,z) =& (x-2)^4 + (x-2y)^2 - sin(z) \\
T3 \quad f(x,y,z) =& e^{(-x-y)} + x^4 + y^2 + 2(y+z-6)^2
\end{aligned}
$$

For the quadratic case consider T1. The exact solution $[1\ 0\ 0]^T$ is the only minimizer. Taking $x_0 = [3\ 3\ 3]^T$ and tolerance $\epsilon = 10^{-4}$, the convergence criteria $\|\nabla f_k\| \leq \epsilon$, we get the following output from MATLAB using Newtons method.

Table 4.1:Newton method with quadratic function

Iter	x	y	z	$\|g\|$	obj
1	1.00000	0.00000	0.00000	3.16228	60.00000
2	1.00000	0.00000	0.00000	0.00000	-1.50000

The conjugate gradient algorithm with the same data inputs and number of iterations to obtain an approximate minimizer $[1\ 0\ 0]^T$.

Table 4.2:Linear conjugate gradient method with quadratic function

k	α	x_1	x_2	x_3	β	$\|r\|$
1	0.277778	0.833333	0.000000	0.277778	0.080247	0.895806
2	0.218692	0.934579	-0.121495	0.149533	0.070749	0.238272
3	0.823077	1.000000	0.000000	0.000000	0.000000	0.000000

However, the steepest descent method with inexact step length and with data inputs $x_0 = [3\ 3\ 3]^T$ and stopping criteria $\epsilon = 10^{-4}$, and Wolfe constants $c_1 = 0.1, c_2 = 0.5$, requires 21 iterations to obtain an approximate minimizer $[0.999984\ \ -0.000018\ \ 0.000037]^T$ of the exact solution $[1\ 0\ 0]^T$,the convergence criteria $\|\nabla f_k\| \leq \epsilon$. Some of the iterations are shown below.

Table 4.3:Steepest descent method with quadratic function

k	x_1	x_2	x_3	$\|g\|$	obj
1	0.750000	0.000000	0.250000	0.866025	-1.375000
2	0.875000	-0.125000	0.125000	0.353553	-1.468750
3	1.000000	0.000000	0.125000	0.467707	-1.476563
4	0.968750	-0.062500	0.031250	0.207289	-1.494141
5	0.984375	-0.015625	0.046875	0.098821	-1.498047
6	0.984375	-0.023438	0.023438	0.052987	-1.499176
7	0.996094	0.000000	0.019531	0.067658	-1.499481
8	0.994141	-0.009766	0.005859	0.030758	-1.499855
18	0.999935	-0.000120	0.000036	0.000480	-1.500000
19	0.999975	-0.000018	0.000085	0.000218	-1.500000
20	0.999972	-0.000043	0.000037	0.000108	-1.500000
21	0.999984	-0.000018	0.000037	0.000060	-1.500000

For the nonquadratic case consider T2. The exact solution $[2 \quad 1 \quad \pi]^T$ is the only minimizer. Taking $x_0 = [0 \quad 0 \quad 0.5\pi]^T$ and tolerance $\epsilon = 10^{-4}$, the convergence criteria $\|\nabla f_k\| \leq \epsilon$,and The approximate minimizer is: $x^* = [1.9846 \quad 0.9923 \quad 1.5708]^T$ we get the following output from **MATLAB** using Newtons method.

Table 4.4:Newton method with nonquadratic function

k	x_1	x_2	x_3	$\|g\|$	obj
1	0.66667	0.33333	1.57080	32.00000	15.00000
2	1.11111	0.55556	1.57080	9.48148	2.16049
3	1.40741	0.70370	1.57080	2.80933	-0.37570
4	1.60494	0.80247	1.57080	0.83239	-0.87668
5	1.73663	0.86831	1.57080	0.24664	-0.97564
6	1.82442	0.91221	1.57080	0.07308	-0.99519
7	1.88294	0.94147	1.57080	0.02165	-0.99905
8	1.92196	0.96098	1.57080	0.00642	-0.99981
9	1.94798	0.97399	1.57080	0.00190	-0.99996
10	1.96532	0.98266	1.57080	0.00056	-0.99999
11	1.97688	0.98844	1.57080	0.00017	-1.00000
12	1.98459	0.99229	1.57080	0.00005	-1.00000

The Fletcher Reeves algorithm with the same data inputs and tolerance $\epsilon = 10^{-4}$, needs 15 iterations to obtain an approximate minimizer $x^* = [2.0000 \quad 1.0000 \quad 1.5708]^T$ and the convergence criteria $\|\nabla f_k\| \leq \epsilon$.

Table 4.5:Fletcher Reeves method with nonquadratic function

k	α	x_1	x_2	x_3	obj	$\|r\|$	β
1	0.2500	1.0000	0	1.5708	1.0000	4.1231	1.0625
2	0.1600	1.1600	0.6400	1.5708	-0.2800	1.8629	0.2041
3	0.1698	1.4656	0.5585	1.5708	-0.5929	1.5693	0.7096
4	0.2306	1.6317	0.8800	1.5708	-0.8479	1.0058	0.4108
5	0.1298	1.7439	0.8134	1.5708	-0.9207	0.6128	0.3713
6	0.3661	1.8885	0.9849	1.5708	-0.9810	0.4452	0.5277
7	0.1111	1.9223	0.9488	1.5708	-0.9934	0.1639	0.1355
8	0.5468	1.9937	1.0029	1.5708	-0.9998	0.0545	0.1105
9	0.1058	1.9963	0.9978	1.5708	-1.0000	0.0072	0.0176
10	0.5402	1.9999	0.9994	1.5708	-1.0000	0.0044	0.3795
11	0.1116	1.9998	0.9999	1.5708	-1.0000	0.0004	0.0080
12	0.3580	1.9999	0.9999	1.5708	-1.0000	0.0001	0.0886
14	0.2262	2.0000	1.0000	1.5708	-1.0000	0.0001	0.9910
15	0.1724	2.0000	1.0000	1.5708	-1.0000	0.0001	0.2478

However, the steepest descent method with inexact step length and with data inputs $x_0 = [0 \quad 0 \quad 0.5\pi]^T$ and tolerance $\epsilon = 10^{-4}$, and Wolfe constants $c_1 = 0.1, c_2 = 0.5$, requires 618 iterations to obtain an approximate minimizer $x^* = [1.9714 \quad 0.9857 \quad 1.5708]^T$ of the exact solution $[2 \quad 1 \quad \pi]^T$ and the convergence criteria $\|\nabla f_k\| \leq \epsilon$. Some of the iterations are shown below.

Table 4.6:Steepest descent method with nonquadratic function

k	t	x_1	x_2	x_3	$\|g\|$	obj
1	0.100000	2.000000	0.000000	1.570796	8.944272	3.000000
2	0.100000	1.500000	1.000000	1.570796	2.500000	-0.687500
3	0.100000	1.687500	0.750000	1.570796	0.791501	-0.955307
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
616	0.100000	1.971389	0.985696	1.570796	0.000100	-0.999999
617	0.100000	1.971439	0.985690	1.570796	0.000235	-0.999999
618	0.100000	1.971436	0.985719	1.570796	0.000100	-0.999999

However, if we consider T2 at $x_0 = [0 \quad 0 \quad 0]^T$, Newtons method not converge. The Fletche-Reeves method and steepest descent methods converges respectively results $[2.0000 \quad 1.0000 \quad 1.5707]^T$ with 33 iterations and $[1.971575 \quad 0.985777 \quad 1.570796]^T$ with 625 iteration respectively and the convergence criteria $\|\nabla f_k\| \leq \epsilon$. Thus, it is better to use the Fletcher -Reeves method instead of both Newtons and the steepest descent method.

Table 4.8:Fletcher Reeves method with nonquadratic function

k	α	x_1	x_2	x_3	obj	$\|r\|$	β
1.	0.2651	1.0603	0 gh8yu	2.8765	1.7453	4.4261	1.1524
2.	0.1572	1.1891	0.6667	2.7248	0.2736	2.0708	0.2189
3.	0.1917	1.5277	0.5560	2.5495	-0.1623	1.9318	0.8703
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
31	0.1264	2.0000	1.0000	1.5710	-1.0000	0.0002	0.1574
32	0.4236	2.0000	1.0000	1.5709	-1.0000	0.0002	1.6108
33	0.1264	2.0000	1.0000	1.5709	-1.0000	0.0001	0.1574

For the nonquadratic case consider T3. The approximated minimizer is $x_0 = [0.4933\ \ 0.2402\ \ 5.7598]^T$ is the only minimizer. Taking $x_0 = [0\ \ 0\ \ 0]^T$ and tolerance $\epsilon = 10^{-4}$, and the convergence criteria $\|\nabla f_k\| \leq \epsilon$, we get the following output from **MATLAB** using conjugate gradient method.

Table 4.7:Fletcher Reeves method with nonquadratic function

k	α	x_1	x_2	x_3	obj	$\|r_k\|$	β
1	0.2117	0.2117	2.7516	2.5399	8.6288	4.2760	0.0582
2	0.2376	0.2149	1.7931	2.8766	6.8912	2.7775	0.4219
3	0.8278	0.2932	1.1381	5.0792	1.6361	2.5135	0.8190
4	0.1592	0.3152	0.7445	5.0100	1.0313	0.8452	0.1131
5	0.5732	0.4420	0.3712	5.2914	0.8471	0.7787	0.8487
6	0.2731	0.4688	0.4738	5.4757	0.6675	0.4687	0.3623
7	0.1800	0.4647	0.3915	5.4939	0.6509	0.2641	0.3174
8	1.1595	0.4917	0.2417	5.7596	0.5971	0.0081	0.0009
9	0.2041	0.4927	0.2405	5.7590	0.5971	0.0022	0.0766
10	0.3152	0.4933	0.2406	5.7593	0.5971	0.0009	0.1735
11	0.1864	0.4933	0.2404	5.7593	0.5971	0.0005	0.3244
12	1.0447	0.4933	0.2402	5.7598	0.5971	0.0002	0.1692
13	0.1702	0.4933	0.2402	5.7598	0.5971	0.0001	70.0743

Therefore, it is better to use the Conjugate gradient method instead of both Newtons and the steepest descent method.

Conclusion

In this Project we have comparing the three line search methods implemented to solve linear and nonlinear equations. The Newton's method is a fast rate of local convergence. After a neighborhood of the solution is reached, convergence to high accuracy often occurs in just a few iterations. The main draw back of Newton method is not guaranteed to produce descent direction when the current iterate is not close to a solution. And the hessian matrix $\nabla^2 f_k$ may not always be positive definite, p_k^N may not always a descent direction.

The steepest descent method is a slow rate of globally convergence. Advantage of the steepest descent direction is that requires the calculation of the gradient ∇f_k but not the second derivative. The main draw back of the steepest descent method is a zigzagging method to converges long iteration.

We can conclude that the conjugate gradient method and its extension alleviate the draw back of the steepest descent method and Newton's method. Numerical comparisons is better to use the Conjugate gradient method and its extension instead of both Newtons and the steepest descent method to solve an unconstrained optimization problems depending on the number of iterations and the number of functions evaluation.

Bibliography

[1] Jorge Nocedal Stephen J. Wright Numerical Optimization Second Edition 2006:(102-131).

[2] E. K. P. Chong and S. H. Zk Fort Collins, Colorado, and West Lafayette, Indiana An Introduction to Optimization Second Edition:48-50.

[3] DavidG.Luenberger and Yinyu Ye Stanford University Linear and Nonlinear Programming Fourth Edition Stanford, California D.G.L. January 2015:277-287.

[4] Wenyu Sun, Nanjing Normal University and Yaxiang Yuan, Chinese Academy of Science Optimization theory and methods Nonlinear Programming April 2005:

[5] Wenyu Sun, Nanjing Normal University and Yaxiang Yuan, Chinese Academy of Science Optimization theory and methods Nonlinear Programming April 2005:

[6] C. T. Kelley and Raleigh, North Carolina Iterative Methods for Optimization.

[7] Salah Gazi Shareef1 and Alaa Luqman Ibrahim A New Conjugate Gradient For Unconstrained Optimization based on step size of Barzilai and Borwein (Accepted for publication: April 11, 2016)

[8] Y.H. Dai Y. Yuan A note on the nonlinear conjugate gradient method Journal of Computational Mathematics, Vol.20, No.6, 2002

[9] Andreas Antoniou Wu-Sheng Lu Department of Electrical and Computer Engineering University of Victoria, Canada PRACTICAL OPTIMIZATION Algorithms and Engineering Applicatio 2007:(145-171)

[10] C. T. Kelley Iterative Methods for Linear and Nonlinear Equations North Carolina State University Society for Industrial and Applied Mathematics Philadelphia 1995.

[11] Andrew R. Conn, Nicholas I. M. Gould and Philippe L. Toint Trust-Region Methods.

Appendix

```
%M-file on Matlab code Table 4.1, Table 4.2, Table 4.3, Table 4.4, Table 4.5,Table 4.6
function  quadratic and nonquadratic euation.
   ### Steepest descent for Matlab code
function steep
 x=input('enter the columen vector x');
   n=length(x);
  a= 0.1;
  b = 0.5;
  obj=z(x);
  g=gr(x);
  k=0;
  nf=1;
  tol=10^(-6);
while  norm(g) > tol
    d = -g;
    t = 1;
    newobj = z(x + t*d);
    nf = nf+1;
 while (newobj-obj)/t > a*g'*d
      t = t*b;
      newobj = z(x + t*d);
      nf = nf+1;

  end
    x = x + t*d;
    obj=newobj;
    g=gr(x);
    k = k + 1;
end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function  z=func(x)
   z=(x(1)-2)^4+(x(1)-2*x(2))^2-sin(x(3));
function y=gr(x)
  y(1)=4*(x(1)-2)^3+2*(x(1)-2*x(2));
  y(2)=-4*(x(1)-2*x(2));
 y(3)=-cos(x(3));
 y=y';
```

```
    ### Newton's method for Matlab code
function Newton
     x=[0;3;1.5*pi];
    n=3;
  obj=z(x)
  g=gr(x);
  H=h(x);
  k=1;
  tol=10^(-6);
 while norm(g)>tol
      obj=z(x);
      g=gr(x);
      H=h(x);
      X=-inv(H)*g;
      x=x+X;
      k=k+1;
 fprintf('%3.0f   %5.5f   %5.5f   %5.5f   %5.5f   %5.5f\n',k-1,x(1),x(2),x(3),norm(H),
end
 disp('The minimizer is:'), x
 disp('The min value: is'), obj
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 function  z=func(x)
   z=(x(1)-2)^4+(x(1)-2*x(2))^2-sin(x(3));
 function y=gr(x)
  y(1)=4*(x(1)-2)^3+2*(x(1)-2*x(2));
  y(2)=-4*(x(1)-2*x(2));
  y(3)=-cos(x(3));
  y=y';
 function h= hes(x)
  h=[12*(x(1)-2)^2+2 -4 0;-4 8 0;0 0 sin(x(3))];

   ### Linear conjugate gradient Matlab code
function [x] = linear conjgrad
k=0;
tol=0.0000001;
 x=input('inter the column vector x');
A=[3 0 1;0 4 2;1 2 3];
b=[3;0;1];
    r = b - A * x;
    p = r;
    rsold = r' * r;
while norm(r)>tol
```

```
        Ap = A * p;
        alpha = rsold / (p' * Ap);
        x = x + alpha * p;
        r = r - alpha * Ap;
        rsnew = r' * r;
        B=rsnew/rsold;
        p = r + (rsnew / rsold) * p;
        rsold = rsnew;
    k=k+1;
end

    ### Nonlinear conjugate gradient method for Matlab code
function nonlinear conjugate gradient line exact
clc;
clear all;
k=0;
x_0=[0,3,1.5*pi]';    %we can change our initial guess here.
tol=10^(-6);
r=grad(x_0(1), x_0(2),x_0(3));
while norm(g)>= tol
     p=-r;          %Compute the search direction
  %To find the step length, Do the exact line search
     h=@(a)f(x_0(1) +a*p(1), x_0(2) +a*p(2),x_0(3) + a*p(3));
    a = fminbnd(h,-2,2);
     x_new=x_0+a*p;           % the new iteration
  rnew=grad(x_new(1), x_new(2),x_new(3));
 \beta=(rnew'*rnew)/(r'*r);
  pnew=-g+b*p;
        x_0=x_0+a*p;
        r=grad(x_new(1), x_new(2),x_new(3));
        k=k+1;
     disp([k,a,x_0',f(x_0(1), x_0(2),x_0(3)),norm(g),b])
end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function w=f(x,y,z)
   w=(x-2)^2 + (x-2*y)^2 -sin(z);
    function r=grad(x,y,z)                   %Gradient of f
        dfdx=2*(x-2) + (x-2*y);
        dfdy=-4*(x-2*y);
        dfdz= -cos(z);
    r=[dfdx;dfdy;dfdz];
```

```
function nonlcg_exact1
k=0;
x_0=[0,0,0]';    %we can change our initial guess here.
tol=10^(-4);
g=grad(x_0(1), x_0(2),x_0(3));
while norm(g)>= tol
      %Compute the search direction
         p=-g;
  %To find the step length, Do the exact line search
     h=@(a)func(x_0(1) +a*p(1), x_0(2) +a*p(2),x_0(3) + a*p(3));
    a = fminbnd(h,-2,2);
     x_new=x_0+a*p;           % the new iteration
 gnew=grad(x_new(1), x_new(2),x_new(3));
 b=(gnew'*gnew)/(g'*g);
pnew=-g+b*p;
        x_0=x_0+a*p;
        g=grad(x_new(1), x_new(2),x_new(3));
        k=k+1;
     disp([k,a,x_0',func(x_0(1), x_0(2),x_0(3)),norm(g),b])
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 function y = func(x,y,z)
y = exp(-x- y) + x.^4+y.^2 +2*(y+z-6).^2;
function g= grad(x,y,z)                     %Gradient of f
dfdx =  -exp(-x-y)+ 4*x.^3 ;
dfdy = -exp(-x-y)+ 2*y+2*(y+z-6);
dfdz = 2*(y+z-6);
 g=[dfdx;dfdy;dfdz];
```