Republic of Iraq
Ministry of Higher Education & Scientific Research
AL-Qadisiyah University
College of Computer Science and Mathematics
Department of Mathematics

Nonlinear Conjugate, Gradient Methods For Unconstrained Optimization

A Research

Submitted by

Hader Ali Ghadban

To the Council of the department of Mathematics/ College of Education, University of AL-Qadisiyah as a Partial Fulfilment of the Requirements for the Bachelor Degree in Mathematics

Supervised by

Alaa Kamel Jaber

A. D. 2017 ***A.H. 1438***

Abstract

Conjugate gradient methods are a class of important methods for solving linear equations and for solving nonlinear optimization. In this article, a review on conjugate gradient methods for unconstrained optimization is given. Conjugate gradient (CG) methods comprise a class of unconstrained optimization algorithms which are characterized by low memory requirements and strong local and global convergence properties.

1. Introduction

Optimization models attempt to express, in mathematical terms, the goal of solving a problem in the "best" way. That might mean running a business to maximize profit, minimize loss, maximize efficiency, or minimize risk. It might mean designing a bridge to minimize weight or maximize strength. It might mean selecting a flight plan for an aircraft to minimize time or fuel use. The desire to solve a problem in an optimal way is so common that optimization models arise in almost every area of application. They have even been used to explain the laws of nature, as in Fermat's derivation of the law of refraction for light. Optimization models have been used for centuries, since their purpose is so appealing. In recent times they have come to be essential, as businesses become larger and more complicated, and as engineering designs become more ambitious. In many circumstances it is no longer possible, or economically feasible, for decisions to be made without the aid of such models. In a large, multinational corporation, for example, a minor percentage improvement in

operations might lead to a multimillion dollar increase in profit, but achieving this improvement might require analyzing all divisions of the corporation, a gargantuan task. Likewise, it would be virtually impossible to design a new computer chip involving millions of transistors without the aid of such models. Such large models, with all the complexity and subtlety that they can represent, would be of little value if they could not be solved. The last few decades have witnessed astonishing improvements in computer hardware and software, and these advances have made optimization models a practical tool in business, science, and engineering. It is now possible to solve problems with thousands or even millions of variables.

There are two types of optimization models, the constrained and unconstrained problems. In this article we will discuss the algorithms which are called conjugate gradient methods (CG) to solve the nonlinear problems of unconstrained optimization type.

2. Basics of Unconstrained Optimization

In this section we begin studying the problem

$$\text{minimize} \quad f(x),$$

where no constraints are placed on the variable $x = (x_1, \ldots, x_n)^T$. Unconstrained problems arise, for example, in data fitting, where the objective function measures the difference between the model and the data. Methods for unconstrained problems are of more general value, though, since they form the foundation for methods used to solve constrained optimization problems. We will derive several optimality conditions for the unconstrained optimization problem. One of these conditions, the "first-order necessary condition," consists of a system of nonlinear equations. Applying Newton's method to this system of

equations will be our fundamental technique for solving unconstrained optimization problems.

2.1 Optimality Conditions

We will derive conditions that are satisfied by solutions to the problem

$$\text{minimize } f(x).$$

The conditions for the problem

$$\text{maximize } f(x)$$

are analogous and will be mentioned in passing.

Let x_* denote a candidate solution to the minimization problem. Even if the global minimizer x_* were given to us, it would be difficult or impossible to confirm that it was indeed the global minimizer. It is easier to look for local minimizers. A local minimizer is a point x_* that satisfies the condition

$$f(x_*) \leq f(x) \text{ for all } x \text{ such that } \|x - x_*\| < \epsilon,$$

where ϵ is some (typically small) positive number whose value may depend on x_*. Similarly defined is a strict local minimizer:

$$f(x_*) < f(x) \text{ for all } x \text{ such that } 0 < \|x - x_*\| < \epsilon.$$

It is possible for a function to have a local minimizer and yet have no global minimizer. It is also possible to have neither global nor local minimizers, to have both global and local minimizers, to have multiple global minimizers, and various other combinations. In this form, these conditions are no more practical than those for a global minimizer, since they too require information about the function at an infinite number of points, and the algorithms will only have information at a finite number of points. However, with additional assumptions on the function f ,

practical optimality conditions can be obtained. To obtain more practical conditions, we assume that the function f is differentiable and that its first and second derivatives are continuous in a neighborhood of the point x_*. Not all the conditions that we derive will require this many derivatives, but it will simplify the discussion if the assumptions do not change from condition to condition. All of these conditions will be derived using Taylor series expansions of f about the point x_*. Suppose that x_* is a local minimizer of f . Consider the Taylor series with remainder term

$$f(x_* + p) = f(x_*) + \nabla f(x_*)^T p + \frac{1}{2} p^T \nabla^2 f(\xi) p,$$

Where p is a nonzero vector and ξ is a point between x and x_*. We will show that

$$\nabla f(x_*) = 0.$$

If x_* is a local minimizer, there can be no feasible descent directions at x_*. Hence

$$\nabla f(x_*)^T p \geq 0$$

for all feasible directions p. For an unconstrained problem, all directions p are feasible, and so the gradient at x_* must be zero. Thus, if x_* is a local minimizer of f , then

$$\nabla f(x*) = 0.$$

A point satisfying this condition is a stationary point of the function f .

In the one-dimensional case, there is a geometric interpretation for this condition. If f is increasing at a point x, then $f'(x) > 0$. Similarly if f is decreasing, then $f'(x) < 0$.

A point where f is increasing or decreasing cannot correspond to a minimizer. At a minimizer the function will be flat or stationary, and hence $f'(x_*) = 0$. This is illustrated in Figure 2.1.

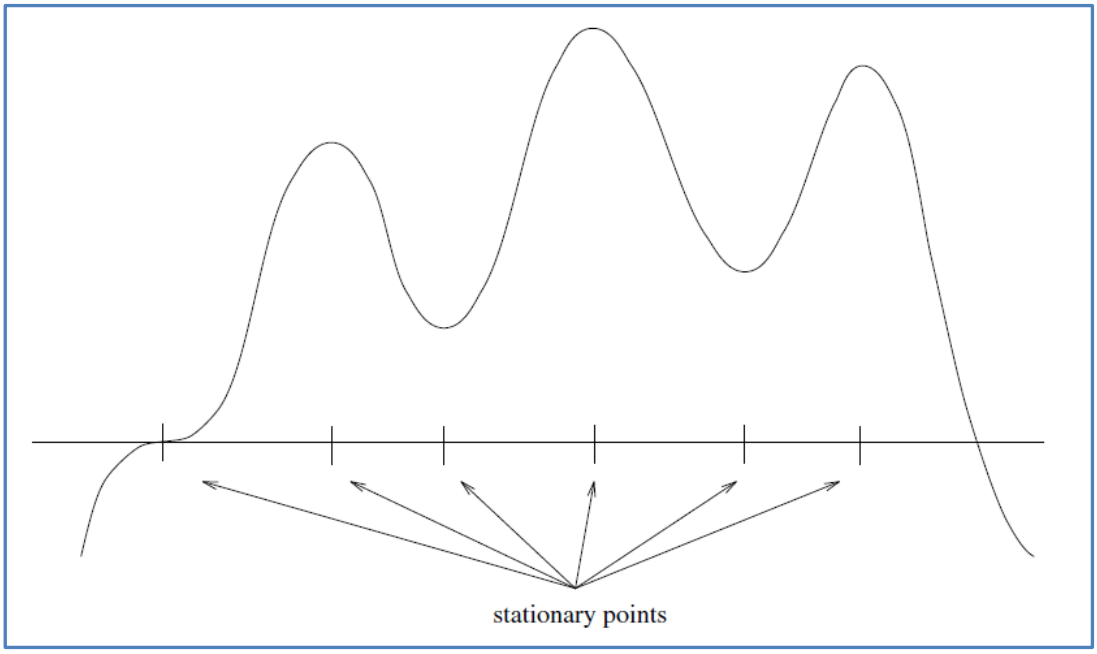

Figure 2.1. *Stationary points*.

The condition $\nabla f(x_*) = 0$ is referred to as the first-order necessary condition for a minimizer. The term "first-order" refers to the presence of the first derivatives of f (or to the use of the first-order term in the Taylor series to derive this condition). It is a "necessary" condition since if x_* is a local minimizer, then it "necessarily" satisfies this condition. The condition is not "sufficient" to determine a local minimizer since a point satisfying x_* could be a local minimizer, a local maximizer, or a saddle point (stationary point that is neither a minimizer nor a maximizer).

Local minimizers can be distinguished from other stationary points by examining second derivatives. Consider again the Taylor series expansion at $x = x_* + p$, but now using the result that

$$\nabla f(x_*) = 0:$$

$$f(x) = f(x_* + p) = f(x_*) + \frac{1}{2} p^T \nabla^2 f(\xi) p,$$

We will show that $\nabla^2 f(x_*)$ must be positive semidefinite. If not, then $v^T \nabla^2 f(x_*) v < 0$ for some v. Then it is also true that $v^T \nabla^2 f(\xi) v < 0$ if $\|\xi - x_*\|$ is small. This is because $\nabla^2 f$ is assumed to be continuous at x_*. If p is

chosen as some sufficiently small multiple of v, then the point ξ will be close enough to x_* to guarantee (via the Taylor series) that $f(x) < f(x_*)$, a contradiction. Hence if x_* is a local minimizer, then $\nabla^2 f(x_*)$ is positive semi definite. This is referred to as the second-order necessary condition for a minimizer, with the "second-order" referring to the use of second derivatives or the second-order term in the Taylor series.

There is also a second-order sufficient condition, "sufficient" to guarantee that x_* is a local minimizer:

If $\nabla f(x_*) = 0$ and $\nabla^2 f(x_*)$ is positive definite,

3. Conjugate Gradient Method

Conjugate gradient methods are a class of important methods for solving unconstrained optimization problem

$$min\, f(x)\,, x \in R^n \tag{3.1}$$

especially if the dimension n is large. They are of the form

$$x_{k+1} = x_k + \alpha_k d_k \tag{3.2}$$

where α_k is a step size obtained by a line search, which will discuss in the following section, and d_k is the search direction defined by

$$d_k = \begin{cases} -g_k & ,for\ k = 1 \\ -g_k + \beta_k d_{k-1}, & for\ k \geq 2 \end{cases} \tag{3.3}$$

where β_k is a parameter, and g_k denotes $\nabla f(x_k)$. It is known from (3.2) and (3.3) that only the step size α_k and the parameter β_k remain to be determined in the definition of conjugate gradient methods. In the case that f is a convex quadratic, the choice of β_k should be such that the method (3.2)-(3.3) reduces to the linear conjugate gradient method if the line search is exact, namely,

$$\alpha_k = arg\ minf(x_k + \alpha d_k,\ \alpha > 0. \qquad (3.4)$$

For nonlinear functions, however, different formulae for the parameter β_k (introduced in table 3.1) 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 $\beta_k$is called conjugate gradient parameter. 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.

$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}$	(1952) Hestenes and Stiefel [5]
$\beta_k^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}$	(1964) Fletcher and Reeves [8]
$\beta_k^{D} = \frac{g_{k+1}^T \nabla^2 f(x_k) d_k}{d_k^T \nabla^2 f(x_k) d_k}$	(1967)
$\beta_k^{PRP} = \frac{g_{k+1}^T y_k}{\|g_k\|^2}$	(1969) Polak and Ribiere [2]and by Polyak[1]
$\beta_k^{CD} = \frac{\|g_{k+1}\|^2}{-d_k^T g_k}$	(1987) proposed by Fletcher [9], CD stands for "Conjugate Descent"
$\beta_k^{LS} = \frac{g_{k+1}^T y_k}{-d_k^T g_k}$	(1991) proposed by Liu and Storey [14]
$\beta_k^{DY} = \frac{\|g_{k+1}\|^2}{-d_k^T y_k}$	(1999) proposed by Dai and Yuan [12]
$\beta_k^{N} = (y_k - 2d_k \frac{\|y_k\|^2}{d_k^T y_k})^2 \frac{g_{k+1}}{d_k^T y_k}$	(2005) proposed by Hager and Zhang [10]

Table 3.1 : Various choices for the CG update parameter

4. Line search [11]

In each CG iteration, the step size k is chosen to yield an approximate minimum for the problem:

$$\min_{\alpha \geq 0} f(x_k + \alpha d_k) \tag{4.1}$$

Since $\alpha \geq 0$, the direction d_k should satisfy the descent condition

$$g_k^T d_k < 0 \tag{4.2}$$

for all $k \geq 0$. If there exists a constant $c > 0$ such that

$$g_k^T d_k < -c \| g_k \|^2 \tag{4.3}$$

for all $k \geq 0$, then the search directions satisfy the sufficient descent condition. The termination conditions for the CG line search are often based on some version of the Wolfe conditions. The standard Wolfe conditions [6, 7] are

$$f(x_k + \alpha d_k) - f(x_k) \leq \delta \alpha_k g_k^T d_k \tag{4.4}$$

$$g_{k+1}^T d_k \geq \sigma g_k^T d_k \tag{4.5}$$

Where d_k is a descent direction and $0 < \delta \leq \sigma < 1$. The strong Wolfe conditions consists of (4.4) and the following strengthened version of (4.5):

$$\left| g_{k+1}^T d_k \right| \leq -\sigma g_k^T d_k \tag{4.6}$$

In the generalized Wolfe conditions [13], the absolute value in (4.6) is replaced by a pair of inequalities:

$$\sigma_1 g_k^T d_k \leq g_{k+1}^T d_k \leq \sigma_2 g_k^T d_k \tag{4.7}$$

where $0 < \delta \leq \sigma < 1$ and $\sigma_2 \geq 0$. The special case $\sigma_1 = \sigma_2 = \sigma$ corresponds to the strong Wolfe conditions. Ideally, we would like to terminate the line search in a CG algorithm when the standard Wolfe conditions are satis ed. For some CG

algorithms, however, stronger versions of the Wolfe conditions are needed to ensure convergence and to enhance stability.

5. The General Optimization Algorithm [3]

More algorithms for solving optimization problems have been proposed than could possibly be discussed in a single book. This has happened in part because optimization problems can come in so many forms, but even for particular problems such as one-variable unconstrained minimization problems, there are many different algorithms that one could use.

Despite this diversity of both algorithms and problems, all of the algorithms have the same general form.

5.1 Algorithm I: General Optimization Algorithm I

1. Specify some initial guess of the solution x_0.

2. For $k = 0, 1, \ldots$

(i) If x_k is optimal, stop.

(ii) Determine x_{k+1}, a new estimate of the solution.

This algorithm is so simple that it almost conveys no information at all. However it is often helpful to keep in mind that we are still working within this simple and general framework. The algorithm suggests that testing for optimality and determining a new point x_{k+1} are separate ideas, but this is usually not true. Often the information obtained from the optimality test is the basis for the computation of the new point. For example, if we are trying to solve the one-dimensional problem without constraints

$$\text{minimize } f\ (x),$$

then the optimality test will often be based on the condition

$$f'(x) = 0.$$

If $f(x_k) \neq 0$, then x_k is not optimal, and the sign and value of $f(x_k)$ indicate whether f is increasing or decreasing at the point x_k, as well as how rapidly f is changing. Such information is valuable in selecting x_{k+1}. Many of our algorithms will have a more specific form.

5.2 Algorithm II: General Optimization Algorithm II

1. Specify some initial guess of the solution x_0.

2. For $k = 0, 1, \ldots$

(i) If x_k is optimal, stop.

(ii) Determine a search direction p_k.

(iii) Determine a step length α_k that leads to an improved estimate of the solution:

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

In this algorithm, p_k is a search direction that we hope points in the general direction of the solution, or that "improves" our solution in some sense. The scalar α_k is a step length that determines the point x_{k+1}, once the search direction p_k has been computed, the step length α_k is found by solving some auxiliary one-dimensional problem; see Figure 5.1

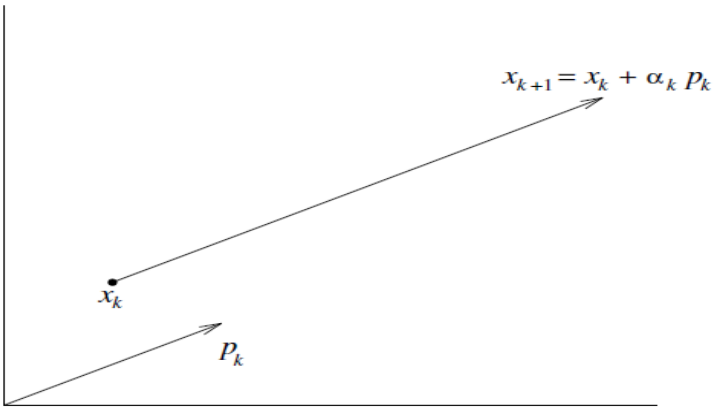

Figure 5.1: *General optimization algorithm.*

Algorithm II with its three major steps (the optimality test, computation of p_k, and computation of α_k) has been the basis for a great many of the most successful optimization algorithms ever developed. It has been used to develop many software packages for nonlinear optimization, and it is also present implicitly as part of the simplex method for linear programming. Our methods are used this general form of algorithm.

5.3 Nonlinear Conjugate-Gradient algorithm

1. Set $p_{-1} = 0, \beta_0 = 0$, and set the convergence tolerance ϵ

2. For $i = 0, 1, \ldots$

(i) If $\|\nabla f(xi)\| < \epsilon$, stop.

(ii) If $i > 0$, set

$$\beta i = \nabla f(x_i)^T \nabla f(x_i) / \nabla f(x_{i-1})^T \nabla f(x_i - 1).$$

(iii) Set $p_i = -\nabla f(x_i) + \beta_i p_{i-1}$.

(iv) Use a line search to determine $x_{i+1} = x_i + \alpha_i p_i$.

This algorithm can be applied to general unconstrained problems:

minimize $f\ (x)$.

It requires the computation of the gradient $\nabla f\ (x)$, but no second-derivative calculations. Its storage requirements are low—just the three vectors x_i , p_i, and $\nabla f\ (x_i)$, plus whatever temporary storage is required by the line search. At each iteration the algorithm requires only a small number of operations on vectors, plus the computation of f and ∇f for various values of x. Hence it is suitable for large problems.

5.3.1 Example 1 We apply the above nonlinear conjugate-gradient method to the minimization problem :

minimize

$$f = \frac{1}{10}[(x_1 - 1)^2 + 2(x_2 - 1)^2 + 3(x_3 - 1)^2 + 4(x_4 - 1)^2] + \left[x_1^2 + x_2^2 + x_3^2 + x_4^2 - \frac{1}{4}\right]^2$$

The initial point is $x_0 = (\ 1\ -1\ 1\ -1\ )^T$

We will use a backtracking line search with parameter $\mu\ =\ 0.1$. We give detailed results for the first two iterations.

At the point x_0 ,

$$f\ (x_0)\ =\ 16.462$$

$$\nabla f\ (x_0)\ =\ (\ 15.0\ -15.8\ 15.0\ -16.6\ )^T\ .$$

At the first iteration the steepest-descent direction is used:

$$p\ =\ -\nabla f\ (x_0)\ =\ (-15.0\ 15.8\ -15.0\ 16.6\ )^T\ .$$

In the line search, the first few trial values of α are rejected:

$$\alpha\ =\ 1$$

$$f\ (x_0 + \alpha p)\ =\ 7.3\ \times\ 10^5$$

$$f\ (x_0)\ +\ \mu\alpha p^T \nabla f\ (x0)\ =\ -81.058$$

.

.

.

$\alpha = 0.0625$

$f(x_0 + \alpha p) = 0.0985$

$$f(x_0) + \mu\alpha p^T \nabla f(x_0) = 10.367$$

but the step length $\alpha = 0.0625$ is accepted. The new point is

$x_1 = (0.0625 \ -0.0125 \ 0.0625 \ 0.0375)^T$

With $f(x_1) = 0.98506$

$\nabla f(x_1) = (-0.24766 \ -0.39297 \ -0.62266 \ -0.80609)^T$.

At this iteration

$\beta_1 = 1.2851 \times 10^{-3}$

and the search direction is

$p = (0.22838 \ 0.41327 \ 0.60338 \ 0.82743)^T$.

In the line search $\alpha = 1$

$f(x_1 + \alpha p) = 1.5712$

$f(x1) + \mu\alpha p^T \nabla f(x_1) = 0.85889$

$\alpha = 0.5$

$$f(x_1 + \alpha p) = 0.46348$$

$f(x_1) + \mu\alpha p^T \nabla f(x_1) = 0.92197$

and the step length $\alpha = 0.5$ is accepted. The new point is

$x_2 = (0.17669 \ 0.19414 \ 0.36419 \ 0.45121)^T$.

This concludes the second iteration.

The complete results are given in Table 5.1. The iteration number is indicated by k. The costs of the method are given by "*ls*" (the number of gradient evaluations).

k	ls	$\|\nabla f\ (x_k)\|$	k	ls	$\|\nabla f\ (x_k)\|$
0	1	2×10^{1}	13	33	$6\times10^{-}5$
1	6	8×10^{-1}	14	35	5×10^{-5}
2	8	2×10^{-1}	15	37	$2\times10^{-}5$
3	11	2×10^{-1}	16	38	2×10^{-5}
4	13	1×10^{-1}	17	41	8×10^{-6}
5	16	6×10^{-2}	18	43	7×10^{-6}
6	18	3×10^{-2}	19	46	2×10^{-6}
7	20	1×10^{-2}	20	48	$9\times10^{-}7$
8	23	1×10^{-3}	21	50	$5\times10^{-}7$
9	24	1×10^{-3}	22	52	1×10^{-7}
10	27	7×10^{-4}	23	54	$1\times10^{-}7$
11	29	5×10^{-4}	24	57	7×10^{-8}
12	32	2×10^{-4}	13	33	$6\times10^{-}5$

Table 5.1. *Nonlinear conjugate-gradient method (n = 4).*

5.2 Example 2: We apply the nonlinear method described in example 5.1 to

$min\ f(x) = \sum_{i=1}^{3}(x_i - 2x_{i+1})^2 = (x_1 - 2x_2)^2 + (x_2 - 2x_3)^2 + (x_3 - 2x_4)^2$

use the initial guess

$$x_0 = (1,1,1,1)^T$$

$f(x) = (x_1 - 2x_2)^2 + (x_2 - 2x_3)^2 + (x_3 - 2x_4)^2$

$f(x_0) = 16.462$

$\nabla f(x_0) = (15.0 \quad 15.8 \quad 15.0 \quad 16.6)^T$

$p = (-15.0 \quad -15.8 \quad -15.0 \quad -16.6)^T$,

$\alpha = 1$

$f(x_0 + \alpha p) = 7.3 \times 10^5$, $f(x_0) + \mu\alpha p^T \nabla f(x_0) = -81.058$

$\alpha = 0.0625$

$f(x_0 + \alpha p) = 0.0985$, $f(x_0) + \mu\alpha p^T \nabla f(x_0) = 10.367$

But the step length $\alpha = 0.0625$ the new point is

$x = (0.0695 \quad 0.0125 \quad 0.0625 \quad 0.0375)$

$f(x_1)$=0.98506

,$\nabla f(x_1) = (0.24766 \quad 0.39297 \quad 0.62266 \quad 0.8060$

$\beta_1 = 1.2851 \times 10^{-3}$

$p = (-0.22838 \quad -0.41327 \quad -0.60338 \quad -0.82743)$

$\alpha = 1$

$f(x_1 + \alpha p) = 0.46348$

$f(x_1) + map^T \nabla f(x_1) = -0.92197$

$x_2 = (0.17669 \quad 0.19414 \quad 0.36419 \quad 0.45121)^T$

6 Rates of Convergence [3]

Many of the algorithms discussed in this book do not find a solution in a finite number of steps. Instead these algorithms compute a sequence of approximate solutions that we hope get closer and closer to a solution. When discussing such an algorithm, the following two questions are often asked:

• Does it converge?

• How fast does it converge?

It is the second question that is the topic of this section.

If an algorithm converges in a finite number of steps, the cost of that algorithm is often measured by counting the number of steps required, or by counting the number of arithmetic operations required. For example, if Gaussian elimination is applied to a system of n linear equations, then it will require about n_3 operations. This cost is referred to as the computational complexity of the algorithm. For many optimization methods, the number of operations or steps required to find an exact solution will be infinite, so some other measure of efficiency must be used. The rate of convergence is one such measure. It describes how quickly the estimates of the solution approach the exact solution. Let us assume that we have a sequence of points x_k converging to a solution x_*. We define the sequence of errors to be $e_k = x_k - x_*$.

Note that

$$\lim_{k\to\infty} e_k = 0$$

We say that the sequence $\{ x_k \}$ converges to x_* with rate r and rate constant Cif

$$\lim_{k\to\infty} \frac{\| e_{k+1} \|}{\| e_k \|^r} = C$$

And $C < \infty$. To understand this idea better, let us look at some examples.

Initially let us assume that we have ideal convergence behavior

$$\| e_{k+1} \| = C \| e_k \|^r \text{ for all } k$$

so that we can avoid having to deal with limits. When $r = 1$ this is referred to as linear convergence

$$\| e_{k+1} \| = C \| e_k \|$$

If $0 < C < 1$, then the norm of the error is reduced by a constant factor at every iteration. If $C > 1$, then the sequence diverges. (What can happen when $C = 1$?) If we choose

$C = 0.1 = 10^{-1}$ and $\|e_0\| = 1$, then the norms of the errors are

$1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7},$

and seven-digit accuracy is obtained in seven iterations, a good result. On the other hand, if $C = 0.99$, then the norms of the errors take on the values 1, 0.99, 0.9801, 0.9703, 0.9606, 0.9510, 0.9415, 0.9321, . . . ,

and it would take about 1600 iterations to reduce the error to 10^{-7}, a less impressive result. If $r = 1$ and $C = 0$, the convergence is called superlinear. Superlinear convergence includes all cases where $r > 1$ since if

$$\lim_{k\to\infty} \frac{\|e_{k+1}\|}{\|e_k\|^r} = C < \infty$$

$$\lim_{k\to\infty} \frac{\|e_{k+1}\|}{\|e_k\|^r} = \lim_{k\to\infty} \frac{\|e_{k+1}\|}{\|e_k\|^r} \|e_k\|^{r-1} = C \times \lim_{k\to\infty} \|e_k\|^{r-1} = 0$$

When $r = 2$, the convergence is called quadratic. As an example, let $r = 2, C = 1$, and $\|e_0\| = 10^{-1}$. Then the sequence of error norms is

10^{-1}–1, 10^{-2}, 10^{-4}, 10^{-8}, and so three iterations are sufficient to achieve seven-digit accuracy. In this form of quadratic convergence the error is squared at each iteration. Another way of saying this is that the number of correct digits in x_k doubles at every iteration. Of course, if the constant $C = 1$, then this is not an accurate statement, but it gives an intuitive sense of the attractions of quadratic convergence rate. For optimization algorithms there is one other important case, and that is when $1 < r < 2$. This is another special case of super linear convergence. This case is important because (a) it is qualitatively similar to quadratic convergence for the precision of common computer calculations, and (b) it can be achieved by algorithms that only compute first derivatives, whereas to achieve quadratic convergence it is often necessary to compute second derivatives as well. To get a sense of what this form of superlinear convergence looks like, let $r = 1.5$, $C = 1$, and $\|e_0\| = 10^{-1}$.

Then the sequence of error norms is 1×10^{-1}, 3×10^{-2}, 6×10^{-3}, 4×10^{-4}, 9×10^{-6}, 3×10^{-8}, and five iterations are required to achieve single-precision accuracy.

6. REFERENCES

[1]. B. T. Polyak, "**The conjugate gradient method in extreme problems**", USSR Comp. Math. Math. Phys., 9 (1969), pp. 94-112.

[2]. E. Polak and G. Ribiere, "**Note sur la convergence de directions conjugees"**, Rev. Francaise Informat Recherche Opertionelle, 3e Annee 16 (1969), pp. 35-43.

[3]. Igor Griva, Stephen G. Nash and Ariela Sofer, "**Linear and Nonlinear Optimization**", SECOND EDITION, George Mason University ,Fairfax, Virginia, 2009.

[4]. Jianguo Zhang, Yunhai Xiao, and Zengxin Wei, "**Nonlinear Conjugate Gradient Methods with Sufficient Descent Condition for Large-Scale Unconstrained Optimization**", Hindawi Publishing Corporation, Mathematical Problems in Engineering, Volume 2009, Article ID 243290, 16 pages.

[5]. M. R. Hestenes and E. L. Stiefel, "**Methods of conjugate gradients for solving linear systems**", J. Research Nat. Bur. Standards, 49 (1952), pp. 409-436.

[6]. P. Wolfe, "**Convergence conditions for ascent methods**", SIAM Review, 11 (1969), pp. 226-235.

[7]. P. Wolfe, "**Convergence conditions for ascent methods. II: Some corrections**", SIAM Review, 13 (1971), pp. 185-188.

[8]. R. Fletcher and C. Reeves, "**Function minimization by conjugate gradients**", Comput. J., 7 (1964), pp. 149-154.
[9]. R. Fletcher, "**Practical Methods of Optimization vol. 1: Unconstrained Optimization**", John Wiley & Sons, New York, 1987.
[10]. W. W. Hager and H. Zhang, "**A new conjugate gradient method with guaranteed descent and an efficient line search"**, November 17, 2003 (to appear in SIAM J. Optim.).
[11]. William W. Hager and Hongchao Zhang, "**A SURVEY OF NONLINEAR CONJUGATE GRADIENT METHODS**", (2005).
[12]. Y. H. Dai and Y. Yuan, "**A nonlinear conjugate gradient method with a strong global convergence property**", SIAM J. Optim., 10 (1999), pp. 177-182.
[13]. Y. H. Dai and Y. Yuan, "**Convergence properties of the Fletcher-Reeves method**", IMA J.
[14]. Y. Liu and C. Storey, "**Efficient generalized conjugate gradient algorithms, Part 1: Theory**", J. Optim. Theory Appl., 69 (1991), pp. 129-137.
[15]. Yu-Hong Dai, "**Nonlinear Conjugate Gradient Methods**", Academy of Mathematics and Systems Science, Chinese Academy of Sciences.
[16]. Zhen-Jun Shi and Jinhua Guo, "**A new algorithm of nonlinear conjugate gradient method with strong convergence**", Computational & Applied Mathematics, Volume 27, N. 1, pp. 93–106, 2008.