



Original Study

A comparison of the convergence rates of Hestenes' conjugate Gram-Schmidt method without derivatives with other numerical optimization methods

Md Nurul Raihen^{1†}

¹Department of Mathematics and Statistics, University of Toledo, Toledo, Ohio, 43606, USA

Communicated by Juan Luis García Guirao; Received: 26.09.2023; Accepted: 12.02.2024; Online: 03.06.2024

Abstract

This article describes an approach known as the conjugate Gram-Schmidt method for estimating gradients and Hessian using function evaluations and difference quotients, and uses the Gram-Schmidt conjugate direction algorithm to minimize functions and compares it to other techniques for solving $\nabla f = 0$. Comparable minimization algorithms are also used to demonstrate convergence rates using quotient and root convergence factors, as described by Ortega and Rheinbolt to determine the optimal minimization technique to obtain results similar to the Newton method, between the Gram-Schmidt approach and other minimizing approaches. A survey of the existing literature in order to compare Hestenes' Gram-Schmidt conjugate direction approach without derivative to other minimization methods is conducted and further analytical and computational details are provided.

Keywords: Conjugate direction methods, numerical optimization, convergence rates, conjugate Gram-Schmidt without derivative.

AMS 2020 codes: 65K10; 65K05; 65D18; 65D05.

1 Introduction

Finding a numerical approximation to the minimum or maximum of a real-valued function f in some interval $[a, b]$ is a typical computational problem. For instance, in order to minimize functions $g(x)$ of multiple variables, many approaches first need to minimize functions of a single variable of the type

$$\alpha(t) = g(x_0 + ts), \quad (1)$$

where x_0 and s are fixed. In practice, the global minimum is usually of interest, not just the local minimum.

For high-dimensional optimization problems in particular, several numerical approaches need substantial computer resources. This can make them insufficient for extremely complicated or huge functions such as Richard's [1] methods for functions with a large number of variables which are impractical.

Similarly, partial derivatives, and in some cases second partial derivatives of f , are used directly by many methods for constrained or unconstrained minimization of f . The classical technique of steepest descent is an example of a derivative method. Finding partial derivatives of f directly is challenging or impossible in many practical problems.

[†]Corresponding author.

Email address: nurul.raihen@gmail.com

Initial condition and algorithm parameter choices, such as gradient descent's step size, can have a significant impact on numerical optimization methods' performance. Deciding on the correct parameters is not always easy [2]. In addition to that the variable metric method was developed by Davidon [3] in 1963 and the conjugate gradient method of Fletcher and Reeves [4], which is slower but requires less storage than the variable metric method, are possibly the most successful methods that use derivatives.

One prominent approach in numerical optimization for solving systems of linear equations and unconstrained optimization problems is Hestenes' Conjugate Gradient method.

The Conjugate Gradient method (CGM), which was first presented by Hestenes and Stiefel [4] in the middle of the 20th century, is where the Hestenes Conjugate Gradient method can be traced back to its roots. This ground-breaking study established the foundations for iterative optimization approaches that make use of conjugate directions in order to minimize quadratic objective functions in an effective manner. In the beginning, the Conjugate Gradient approach was primarily concerned with solving linear systems of equations; nevertheless, it had several drawbacks when it came to dealing with non-linear optimization issues.

The Hestenes Conjugate Gradient method, also known as the Hestenes-Stiefel conjugate gradient method, is an optimization algorithm used to find the minimum of a function, particularly in the context of solving linear systems of equations or quadratic optimization problems. It is an extension of the more well-known Conjugate Gradient method and is often used in the context of solving linear systems of equations $Ax = b$, where A is a symmetric and positive-definite matrix. Within our analysis, the Gram-Schmidt conjugate direction algorithm is the most efficient. However, it is critical to understand that its performance is greatly dependent on the settings used for difference quotients and scaling.

In 1952, Hestenes and Stiefel pioneered conjugate direction (CD) methods, aiming to minimize a quadratic function in a finite-dimensional space, primarily to enhance computational methods for solving large systems of linear equations [5]. This technique, initially focused on quadratic functions, was expanded to non-quadratic functions by Fletcher and Reeves in 1964, evolving the conjugate gradient (CG) method [6]. Building on this foundation, Nocedal investigated the potential for nonlinear conjugate gradient methods to achieve convergence without resets and using effective line searches [7, 8]. The applications and implications of conjugate direction methods in numerical optimization have been further explored by various researchers, including Kelley [9], Zang and Li [10], among others, showcasing the breadth and adaptability of these methods in the field of numerical optimization.

The Hestenes Conjugate Gradient method is particularly useful for solving large-scale linear systems with symmetric and positive-definite matrices. CGM is an algorithm that uses relatively little memory. Few vectors of the same dimension as the problem size are all that's needed for storage. In contrast, approaches such as the Gauss-Newton or BFGS methods necessitate the storage of matrices that, when applied to high-dimensional problems, might become exceedingly huge [11]. CGM combines the advantages of the Conjugate Gradient method with the Hestenes-Stiefel method's improved convergence properties. It aims to minimize the number of iterations needed to find a solution while avoiding some of the issues that can arise with the standard Conjugate Gradient method, especially for ill-conditioned or non-symmetric matrices [12].

In order to acquire a comprehensive understanding of the Gram-Schmidt method's effectiveness, our research provides a comprehensive comparison analysis. Our evaluation of the convergence rates of various minimization strategies is accomplished through the utilization of a combination of analytical and computational methodologies [13]. We intend to provide quantitative evaluations of the performance of a number of different algorithms by making use of the definitions of quotient convergence factors and root convergence factors that were presented by Ortega and Rheinbolt [14]. Our primary focus is on carrying out a comprehensive analysis of the Gram-Schmidt conjugate direction algorithm. In this analysis, we evaluate the effectiveness of this algorithm in comparison to other well-established methods.

Our main objective in this study is to apply Hestenes' Gram-Schmidt conjugate direction (GSCD) method without derivatives and to compare it to other numerical optimization methods. There is no need to minimize functions or take precise gradient measurements when using the Gram-Schmidt conjugate direction method that

Hestenes devised [15]. Although Hestenes calls it the Conjugate Gram-Schmidt (CGS) method, from here on we'll call it the GSCD method.

The GSCD technique can be numerically demonstrated and compared to other methods by computing Ortega and Rheinboldt's asymptotic constants and quotient convergent factors [14]. This was done to demonstrate that the GSCD methodology is superior to other methods. According to Hestenes [15] (in page 202), the Conjugate Gradient Squared has Newton's algorithm as its limit, which is supported computationally by comparing the numerical results of different approaches. The GSCD algorithm's quadratic convergence is numerically verified under acceptable conditions. However, Russak [16] shows the potential of n -step super-linear convergence.

To denote matrices in the formula, we use uppercase letters like $A, B, C, \dots$, and lowercase letters like $a, b, c, \dots$, while scalars are only expressed by using lowercase letters. In this context, A^* represents the matrix inversion of A . If F is a real-valued differentiable function of n real variables, the gradient at x is denoted as $F'(x)$, and the Hessian of F at x is indicated as $F''(x)$. For example, $x_k = (x_k^1, \dots, x_k^n)$ uses subscripts to distinguish vectors and superscripts to indicate components.

The rest of this paper is organized as follows: Section 2 discusses the methodology, which uses the well-known conjugate directions approach for minimizing a quadratic function modified to become an algorithm for minimizing a nonquadratic function. Section 3 details the algorithms used for each minimization method. Section 4 analyses and discusses the numerical results. Section 5 demonstrates an instance of applying the adapted conjugate direction method to a nonquadratic function without using derivatives. Section 6 presents a discussion. When compared to Newton's method, the CGS technique has the advantage of being both time and computationally efficient, while other methods don't have it. The findings are summarized in Section 7.

2 Methodology

In this section, we delve into the CGS method, a sophisticated approach within the family of CD algorithms, specifically designed for efficiently minimizing functions in an n -dimensional space.

For more details, the reader is referred to see Stein [17], Raihen [18], and Hestenes [15].

2.1 The CD method

Let A be a $n \times n$ matrix with real entries. We consider that the matrix A is a positive definite and symmetric. Let b be a constant real n -vector and let c be a fixed real scalar. Let G denote the function on S_n by

$$G(x) = \frac{1}{2}x^*Ax - b^*x + c, \quad (2)$$

where x is in S_n , Euclidean n -space.

Definition 1. u and v , two vectors are said to be conjugate or A -orthogonal if $u^*Av = 0$. A set of vectors, $\{u_1, \dots, u_k\}$ is said to be mutually conjugate or mutually A -orthogonal in S_n if

$$u_i^*Au_j = 0, \text{ for } i \neq j, \quad (3)$$

where $i, j = 1, 2, \dots, k$.

Let L_k be the linear subspace and spanned by the set of k linearly independent and non-zero vectors $u_1, \dots, u_k$ for all $k = 1, \dots, n$. Let x_1 be a vector in the space S_n . Then, the k -dimensional plane P_k through x_1 is obtained by translating the subspace S_m , which is defined by

$$P_k = \left\{ x : x = x_1 + \beta_1 p_1 + \dots + \beta_k p_k, \beta_i \in \mathbb{R} (i = 1, \dots, k) \right\}.$$

The following theorem is to find a unique vector in the k -dimensional plane P_k so that the unique vector minimizes F given by (3).

Theorem 1. [12] Let L_k be a subspace of S_n , where $\{p_1, \dots, p_k\}$ is a basis for S_k , $1 \leq k \leq n$. Further assume that $p_1, \dots, p_k$ is a mutually A-orthogonal set of vectors. Let x_1 be any vector in S_n . Let x be in P_k . Then, the following conditions are equivalent:

1. x minimizes G on P_k .
2. $G'(x)$, the gradient of G at x , is orthogonal to the subspace S_k .
3. $x = x_1 + a_1 p_1 + \dots + a_k p_k$, where $a_i = \frac{c_i}{d_i}$, $c_i = -p_i^* G'(x_1)$, $d_i = p_i^* A p_i$, $i = 1, \dots, m$.

Let $x_i = x_1 + a_1 p_1 + \dots + a_i p_i$, $i = 1, \dots, m$. Then the quantity c_i defined in (3) above is also given by

$$c_i = -p_i^* G'(x_i), \quad i = 1, \dots, m.$$

Then, there is a unique vector x_0 in the k -dimensional plane P_k through x_1 translating S_k such that x_0 minimizes the function F given by (3) on P_k .

Now we will look at some examples of conjugate direction approaches. These examples are connected to Theorem (1), which demonstrates multiple ways for finding the conjugate direction vectors P_i . For this purpose, we make use of the quadratic function.

Example 1.

$$G(x^1, x^2) = \frac{(x^1)^2}{8} + \frac{(x^2)^2}{4}. \quad (4)$$

This function can be written as (2) with $n = 2, k = 0, c = 0$ and the matrix $A = \begin{bmatrix} \frac{1}{8} & 0 \\ 0 & \frac{1}{4} \end{bmatrix}$. In the following case, we will also assume that the initial vector x_1 is $(x_1^1, x_1^2) = (2, \sqrt{2})$, where $(0, 0)$ is the minimizing vector.

At any point $x = (x^1, x^2)$, the gradient of G is given by

$$G'(x) = \left(\frac{x^1}{4}, \frac{x^2}{2} \right).$$

Take $p_1 = -G'(x_1)$, where G is denoted by the equation (4). Therefore, $p_1 = (-\frac{1}{2}, -\frac{\sqrt{2}}{2})$. We obtain the minimizing scalar, $a_1 = \frac{c_1}{d_1} = \frac{\frac{3}{4}}{\frac{5}{64}} = \frac{48}{5}$.

Now set $x_2 = x_1 + a_1 p_1$, then $x_2 = (2, \sqrt{2}) + \frac{48}{5}(-\frac{1}{2}, -\frac{\sqrt{2}}{2}) = (2, \sqrt{2}) + (-\frac{24}{5}, -\frac{24\sqrt{2}}{5}) = (-\frac{14}{5}, -\frac{19\sqrt{2}}{5})$. The next conjugate direction vector p_2 is computed as follows: First compute $G'(x_2)$, we find $G'(x_2) = (-\frac{7}{10}, -\frac{19\sqrt{2}}{10})$. Then compute b_1 , as $b_1 = \frac{\|F'(x_2)\|^2}{\|F'(x_1)\|^2} = \frac{82}{15}$. Finally, we set $p_2 = -G'(x_2) + b_1 p_1 = (\frac{-61}{30}, \frac{-5}{3\sqrt{2}})$.

Example 2. Suppose we start with a set of vectors $\{v_1, v_2\}$ that are linearly independent, and then we use the Gram-Schmidt process to build a set of vectors $\{p_1, p_2\}$ that are orthogonal to the inner product $\langle x, y \rangle = \langle Ax, y \rangle$. let $v_1 = (-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$ and $v_2 = (-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$.

Now Consider $p_1 = v_1$. Then the function is provided by (4) with $x_1 = (\sqrt{2}, 1)$ and we get $a_1 = \frac{2(\sqrt{2}+1)}{3}$ by minimizing the function $F(x_1 + ap_1)$ for all real a . Set $x_2 = x_1 + a_1 p_1$ and hence $x_2 = (\frac{2(\sqrt{2}-1)}{3}, \frac{(1-\sqrt{2})}{3})$.

Therefore, calculating the next conjugate direction vector p_2 is as follows: First, set $\bar{x}_1 = x_2 + u_2$ such that $\bar{x}_1 = (\frac{1-2\sqrt{2}}{3\sqrt{2}}, \frac{\sqrt{2}+1}{3\sqrt{2}})$. Then we obtain $\bar{a}_1$ such that $a = \bar{a}_1$ minimizes $F(\bar{x}_1 + ap_1)$. Continuing this process yields $x_3 = 0$, which is the minimization vector.

2.2 Non-quadratic function minimization using conjugate-direction algorithms

The conjugate direction algorithms used for minimizing a quadratic function can be expanded in a variety of ways to minimize a non-quadratic function. Two algorithms will be assessed both computationally and analytically. Section 3 describes the algorithms extensively.

A GSCD-I algorithm is a first method for extending minimization to a non-quadratic function. It uses exact gradient evaluations and difference quotients for the evaluation of the Hessian. From (2) the quadratic function G , we find

$$r(x) = -G'(x) = k - Ax,$$

and

$$s(p) = \frac{G'(x + \sigma p) - G'(x)}{\sigma} = Ap,$$

for any x and $\sigma \neq 0$, where G' denotes the gradient of G at x . This procedure is a Gram-Schmidt conjugate direction method using only gradient evaluations and is referred to as GSCD below. GSCD Algorithm [12, 13]:

Starting routine. We choose an initial point x_1 and set $p_1 = u_1$, where $u_1 \neq 0$. Compute $f'(x_1)$ and $G'(x_1 + \sigma_1 p_1)$ for σ_1 small. Set

$$s_1 = \frac{G'(x_1 + \sigma_1 p_1) - G'(x_1)}{\sigma_1}, \quad d_1 = s_1^* p_1, \quad a_1 = \frac{-p_1^* G'(x_1)}{d_1}.$$

General routine. We choose u_k to be linearly independent of $u_1, \dots, u_{k-1}$. Having obtained the vectors $p_1, \dots, p_{k-1}, s_1, \dots, s_{k-1}, G'(x_1)$ and the scalars $d_1, \dots, d_{k-1}$ obtain p_k, s_k, d_k , and a_k as follows:

For $k = 2, \dots, n$, we set

$$p_k = u_k - \frac{s_1^* u_k}{d_1} p_1 - \frac{s_2^* u_k}{d_2} p_2 - \dots - \frac{s_{k-1}^* u_k}{d_{k-1}} p_{k-1}.$$

We compute $G'(x_1 + \sigma_k p_k)$ for σ_k small. Set

$$s_k = \frac{G'(x_1 + \sigma_k p_k) - G'(x_1)}{\sigma_k}, \quad d_k = s_k^* p_k, \quad a_k = -\frac{p_k^* G'(x_1)}{d_k}.$$

Finally, we set

$$x_{n+1} = x_1 + a_1 p_1 + a_2 p_2 + \dots + a_n p_n.$$

We reiterate the entire procedure using x_{n+1} in the place of x_1 and continue with this cycle.

A GSCD-II algorithm is a second method for extending minimization to a non-quadratic function. Neither function minimization nor precise evaluation of gradients is needed for this algorithm. It uses function evaluations only. Dennemeyer and Mookini's [19] algorithm is used in our computer program. However, the method given by Hestenes [15] is shown below to be the same. Dennemeyer and Hestenes both call this process a Conjugate Gram-Schmidt algorithm; hence, we refer to the process as CGS.

Initial step. We select an initial point x_1 and a set of linearly independent unit vectors $u_1, \dots, u_n$. We select a small constant $\sigma > 0$ and choose the scale factor $\rho > 0$, e.g., choose σ to be 10^{-3} and choose $\rho = 2\sigma$. We set

$$p_1 = u_1, \quad \gamma_0 = 0.$$

Then, we compute

$$\begin{aligned} c_1 &= \frac{G(x_1 - \sigma p_1) - G(x_1 + \sigma p_1)}{2\sigma}, \\ d_1 &= \frac{G(x_1 - \sigma p_1) - 2G(x_1) + G(x_1 + \sigma p_1)}{\sigma^2}, \\ \gamma_1 &= |c_1|, \quad a_1 = \frac{c_1}{d_1}, \quad x_2 = x_1 + a_1 p_1. \end{aligned}$$

Then, we compute

$$\begin{aligned} c_{21} &= \frac{G(x_1 + \rho u_2 - \sigma p_1) - G(x_1 + \rho u_2 + \sigma p_1)}{2\sigma}, \\ a_{21} &= \frac{c_{21}}{d_1}, \quad b_{21} = \frac{a_{21} - a_1}{\rho}, \\ \bar{p}_2 &= u_2 + b_{21}p_1, \quad p_2 = \frac{\bar{p}_2}{\|\bar{p}_2\|}. \end{aligned}$$

We iterate the steps by giving $p_1, \dots, p_k, d_1, \dots, d_{k-1}, a_1, \dots, a_{k-1}, \gamma_{k-1}$, for $k = 2, 3, \dots, n$. So, we compute

$$\begin{aligned} c_k &= \frac{G(x_k - \sigma p_k) - G(x_k + \sigma p_k)}{2\sigma}, \\ d_k &= \frac{G(x_1 - \sigma p_k) - 2G(x_1) + F(x_1 + \sigma p_k)}{\sigma^2}, \\ \gamma_k &= \max[\gamma_{k-1}, |c_k|], \quad a_k = \frac{c_k}{d_k}, \quad x_{k+1} = x_k + a_k p_k. \end{aligned}$$

For $j = 1, \dots, k$, we compute

$$\begin{aligned} c_{k+1,j} &= \frac{G(x_1 + \rho u_{k+1} - \sigma p_j) - G(x_1 + \rho u_{k+1} + \sigma p_j)}{2\sigma}, \\ a_{k+1,j} &= \frac{c_{k+1,j}}{d_j}, \quad b_{k+1,j} = \frac{a_{k+1,j} - a_j}{\rho}. \end{aligned}$$

Then, we calculate

$$\begin{aligned} \bar{p}_{k+1} &= u_{k+1} + b_{k+1,1}p_1 + \dots + b_{k+1,k}p_k, \\ p_{k+1} &= \frac{\bar{p}_{k+1}}{\|\bar{p}_{k+1}\|}. \end{aligned}$$

The cycle is stopped when k is sufficiently large. We set $x_1 = x_{k+1}$, and repeat the cycle. If at the end of a cycle, γ_k is sufficiently small, then accept x_{k+1} as the solution for the minimum. The iterative steps described above are essentially those of Hestens [15]. Below we replace some of these steps with those described by Dennemeyer's [19]. The computations of $c_{k+1,j}, a_{k+1,j}, b_{k+1,j}, \bar{p}_{k+1}$, and p_{k+1} are carried out as follows:

For each $q = k+1, \dots, n$, calculate $c_{qk} = \frac{G(x_1 + \rho u_q - \sigma p_k) - G(x_1 + \rho u_q + \sigma p_k)}{2\sigma}$, and $a_{qk} = \frac{c_{qk}}{d_k}$, $b_{qk} = \frac{(a_{qk} - a_k)}{\rho}$. Then compute $\bar{p}_{k+1}$ and p_{k+1} as before.

2.3 Rates of convergence

A precise formulation of the asymptotic rate of convergence of a sequence x^k converging to x^* is motivated by the fact that estimates of the form

$$\|x^{k+1} - x^*\| \leq \|x^k - x^*\|^p, \quad (5)$$

for all $k = 1, 2, \dots$, often arise naturally in the study of certain iterative processes.

Definition 2. Let x^k be a sequence of points in R^n that converges to a point x^* . Let $1 \leq p < \infty$. Ortega and Rheinboldt [20] define the quantities

$$\mathcal{Q}_p\{x^k\} = \begin{cases} 0 & \text{if } x^k = x^* \text{ for all but finitely many } k, \\ \limsup_{k \rightarrow \infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^p} & \text{if } x^k \neq x^* \text{ for all but finitely many } k, \\ +\infty & \text{otherwise,} \end{cases}$$

and refer to them as quotient convergence factors, or Q -factors for short.

Definition 3. Let $C(\mathcal{I}, x^*)$ denote the set of all sequences having a limit of x^* that are generated by an iterative process $\mathcal{I}$.

$$\mathcal{Q}_p(\mathcal{I}, x^*) = \sup\{\mathcal{Q}_p\{x^k\} \mid \{x^k\} \in C(\mathcal{I}, x^*)\} \quad 1 \leq p < +\infty,$$

are the $\mathcal{Q}$ -factors of $\mathcal{I}$ at x^* with respect to the norm in which the $\mathcal{Q}_p\{x^k\}$ are computed.

Note that if $\mathcal{Q}_p\{x_k\} < +\infty$ for some p where $1 \leq p < \infty$, then, for any $\varepsilon > 0$, there is some positive integer K such that (5) above holds for $C = \mathcal{Q}_p\{x_k\} + \varepsilon$. If $0 < \mathcal{Q}_p\{x_k\} < \infty$, then we say that x^k converges to x^* with Q -order of convergence p , and if $\mathcal{Q}_p\{x_k\} = 0$, for some fixed p satisfying $1 \leq p < \infty$, then we say that x^k has super-convergence of Q -order p to x^* . For example, if $0 < \mathcal{Q}_p\{x_k\} < +\infty$ when $p = 1$, then we also have $0 < C < 1$ in (5), we say that $\{x_n\}$ converges to x^* linearly. In addition, if $\mathcal{Q}_p\{x_k\} = 0$ when $p = 1$, we say that $\{x_n\}$ converges to x^* super-linearly.

Definition 4. One other method of describing convergence rate involves the root convergence factors [15],

$$\mathcal{R}_p(x_n) = \begin{cases} \limsup_{k \rightarrow \infty} \|x_n - x^*\|^{1/n} & \text{if } p = 1, \\ \limsup_{k \rightarrow \infty} \|x_n - x^*\|^{1/p^n} & \text{if } p > 1. \end{cases}$$

2.4 Acceleration

One acceleration procedure is the following: first, apply n CD steps to an initial point x_1 to obtain a point $x_{n+1} = y_1$; then, take x_{n+1} to be a new initial point and apply n CD steps again to obtain another $x_{n+1} = y_2$; finally, check for acceleration by evaluating $Q = G(y_2 - (Y_2 - y_1))$, if $Q < G(y_2)$; then, we accelerate by taking $[y_2 - (y_2 - y_1)]$ as our initial point; if $Q > F(y_2)$, then take y_2 as a new initial point; after two more applications of the CD method, we check for acceleration again. The procedure continues in this manner [17, 21].

The iterative procedures that were just explained can be applied to a wide variety of problems. They are used in the process of least squares fitting, in the resolution of linear and nonlinear systems of equations, and in the resolution of optimization problems with and without constraints [17]. In recent years, a significant amount of attention has been directed toward the computational performances as well as the numerical findings obtained from comparing and contrasting these various methods. This interest has been concentrated on large-scale applications and the solution of unconstrained optimization problems [22, 23].

3 Algorithms for each optimization technique

In the ensuing section, we delve into a detailed exploration of various algorithms, with a primary focus on the CGS method. This examination also extends to other significant techniques such as the Newton method, the Runge-Kutta method, and the Steepest Descent method [8, 15, 19]. Each of these methods offers unique approaches to solve the optimization problems, and their comparative analysis provides valuable insights into their respective efficacies.

Particularly noteworthy is the application of the CGS method. Renowned for its efficiency in handling large-scale problems, the CGS method stands out due to its accelerated convergence rate, especially when dealing with sparse systems. This aspect is crucial in computational settings where resource optimization is key.

Furthermore, we employ Maple programs to determine the asymptotic constant, a critical factor in assessing the convergence of non-quadratic functions [18]. The asymptotic constant serves as a benchmark to gauge the performance of an algorithm, especially in scenarios where traditional convergence criteria might be inadequate or misleading. It is essential to emphasize the fact that, in comparison to several alternative approaches, the CGS algorithm possesses much better performance.

Algorithm 1 Runge-Kutta Method

```

Step 1: restart: Digits:=50;
Step 2:  $f[1] := (x, y) \rightarrow 2 * (x - 1) - 400 * x(y - x * x)$ ;
Step 3:  $f[2] := (x, y) \rightarrow 200 * (y - x * x)$ ;
Step 4:  $h := 0.001$ 
Step 5:  $t[1] := 0$ ;  $x1[1] := -1.2$ ;  $x2[1] := 1.0$ ;
Step 6: Perform the following iteration:
for  $i$  from 1 to 100 do
    for  $j$  from 1 to 2 do
 $k1[j] := f[j](x1[i], x2[i])$ 
    end for:
    for  $j$  from 1 to 2 do
 $k2[j] := f[j](x1[i] + (h/2) * k1[1], x2[i] + (h/2) * k1[2])$ 
    end for:
    for  $j$  from 1 to 2 do
 $k3[j] := f[j](x1[i] + (h/2) * k2[1], x2[i] + (h/2) * k2[2])$ 
    end for:
    for  $j$  from 1 to 2 do
 $k4[j] := f[j](x1[i] + h * k3[1], x2[i] + h * k3[2])$ 
    end for:  $x1[i+1] := x1[i] + (h/6) * (k1[1] + 2 * k2[1] + 2 * k3[1] + k4[1])$ ;
 $x2[i+1] := x2[i] + (h/6) * (k1[2] + 2 * k2[2] + 2 * k3[2] + k4[2])$ ;
 $t[i+1] := t[i] + h$ 
    end for:
Step 7: print:
for  $i$  from 1 by 10 to 100 do print( $t[i], x1[i], x2[i]$ );
end for;

```

Algorithm 2 Newton method

```

Step 1: restart: Digits:=150;
Step 2: with(Student[LinearAlgebra]):
Step 3: Perform the following iteration:
 $x := \langle -1.2, 1 \rangle$ :
for  $i$  from 1 to 9 do
 $Jacobian := \langle -400 * (x[2] - x[1]^2) + 800 * x[1]^2 + 2, -400 * x[1], 200 \rangle$ :
 $Jacobian^{(-1)}$ :
 $x := x - (Jacobian^{(-1)}) . \langle -400 * x[1] * x[2] + 400 * x[1]^3 + 2 * (x[1] - 1), 200 * (x[2] - x[1]^2) \rangle$ :
print( $x[1], x[2]$ ):
 $y[i, 1] := x[1]$ ;  $y[i, 2] := x[2]$ ;
end for

```

Algorithm 3 Steepest Descent method

Step 1: restart: Digits:=50;
 Step 2: $\text{gradient}[1] := (x, y) - > 2 * (x - 1) - 400 * x(y - x * x);$
 Step 3: $\text{gradient}[2] := (x, y) - > 200 * (y - x * x);$
 Step 4: $\text{NORMofGRADIENT} := \sqrt{(2 * (x - 1) - 400 * x * (y - x * x))^2 + (200 * (y - x * x))^2};$
 Step 5: $\text{NORMALIZED}[1] := (x, y) - > \frac{(2 * (1 - x) + 400 * x * (y - x * x))}{\sqrt{(2 * (x - 1) - 400 * x * (y - x * x))^2 + (200 * (y - x * x))^2}};$
 Step 6: $\text{NORMALIZED}[2] := (x, y) - > \frac{200 * (x * x - y)}{\sqrt{(2 * (x - 1) - 400 * x * (y - x * x))^2 + (200 * (y - x * x))^2}};$
 Step 7: $f[1] := (x, y) - > \frac{(2 * (1 - x) + 400 * x * (y - x * x))}{\sqrt{(2 * (x - 1) - 400 * x * (y - x * x))^2 + (200 * (y - x * x))^2}};$
 Step 8: $f[2] := (x, y) - > \frac{200 * (x * x - y)}{\sqrt{(2 * (x - 1) - 400 * x * (y - x * x))^2 + (200 * (y - x * x))^2}};$
 Step 9: $h := 0.005$
 Step 10: $t[1] := 0; x1[1] := -1.2; x2[1] := 1.0;$
 Step 11: Perform the following iteration:
for i from 1 to 100 **do**
 for j from 1 to 2 **do**
 $k1[j] := f[j](x1[i], x2[i])$
 end for:
 for j from 1 to 2 **do**
 $k2[j] := f[j](x1[i] + (h/2) * k1[1], x2[i] + (h/2) * k1[2])$
 end for:
 for j from 1 to 2 **do**
 $k3[j] := f[j](x1[i] + (h/2) * k2[1], x2[i] + (h/2) * k2[2])$
 end for:
 for j from 1 to 2 **do**
 $k4[j] := f[j](x1[i] + h * k3[1], x2[i] + h * k3[2])$
 end for: $x1[i + 1] := x1[i] + (h/6) * (k1[1] + 2 * k2[1] + 2 * k3[1] + k4[1]);$
 $x2[i + 1] := x2[i] + (h/6) * (k1[2] + 2 * k2[2] + 2 * k3[2] + k4[2]);$
 $t[i + 1] := t[i] + h$
end for:
 Step 12: print:
for i from 1 by 10 to 100 **do** $\text{print}(t[i], x1[i], x2[i]);$
end for;

Algorithm 4 CGS algorithm without derivative

```

Step 1: restart: Digits:=400;
Step 2:  $\sigma := 1e^{-120}$ ;  $\rho := 2 * \sigma$ ;  $\epsilon := .1e^{-60}$ ;
Step 3:  $u[1, 1] := 1$ ;  $u[1, 2] := 0$ ;  $u[2, 1] := 0$ ;  $u[2, 2] := 1$ ;
Step 4:  $p[1, 1] := u[1, 1]$ ;  $p[1, 2] := u[1, 2]$ ;
Step 5:  $f := (x_1, x_2) \rightarrow 100 * (x_2 - x_1)^2 + (x_1 - 1)^2$ ;
Step 6:  $x[1, 1] := -1.2$ ;  $x[1, 2] := 1$ ;
Step 7: Perform the following iteration:
for  $j$  from 1 to 10 do
   $c[1] := (f(x[1, 1]) - \sigma * p[1, 1], x[1, 2] - \sigma * p[1, 2]) - f(x[1, 1] + \sigma * p[1, 1], x[1, 2] + \sigma * p[1, 2]) / (2 * \sigma)$ ;
   $d[1] := (f(x[1, 1] - \sigma * p[1, 1], x[1, 2] - \sigma * p[1, 2]) - 2 * f(x[1, 1], x[1, 2]) + f(x[1, 1] + \sigma * p[1, 1], x[1, 2] + \sigma * p[1, 2])) / (\sigma)^2$ ;
   $a[1] := c[1] / d[1]$ ;
   $x[2, 1] := x[1, 1] + a[1] * p[1, 1]$ ;
   $x[2, 2] := x[1, 2] + a[1] * p[1, 2]$ ;
  for  $i$  from 1 to 2 do
     $x[2, i] := x[1, i] + a[1, i]$ 
  end for;
   $c[2, 1] := (f(x[1, 1] + \rho * u[2, 1] - \sigma * p[1, 1], x[1, 2] + \rho * u[2, 2] - \sigma * p[1, 2]) - f(x[1, 1] + \rho * u[2, 1] + \sigma * p[1, 1], x[1, 2] + \rho * u[2, 2] + \sigma * p[1, 2])) / (2 * \sigma)$ ;
   $a[2, 1] := c[2, 1] / d[1]$ ;
   $b[2, 1] := (a[2, 1] - a[1]) / \rho$ ;
  for  $i$  from 1 to 2 do
     $pbar[2, i] := u[2, i] + b[2, 1] * p[1, i]$ 
  end for;
  for  $i$  from 1 to 2 do
     $p[2, i] := pbar[2, i] / \sqrt{(pbar[2, 1])^2 + (pbar[2, 2])^2}$ 
  end for;
  for  $k$  from 2 to 2 do
     $c[k] := (f(x[1, 1] - \sigma * p[k, 1], x[1, 2] - \sigma * p[k, 2]) - f(x[1, 1] + \sigma * p[k, 1], x[1, 2] + \sigma * p[k, 2])) / (2 * \sigma)$ ;
  end for;
  for  $k$  from 2 to 2 do
     $d[k] := (f(x[1, 1] - \sigma * p[k, 1], x[1, 2] - \sigma * p[k, 2]) - 2 * f(x[1, 1], x[1, 2]) + f(x[1, 1] + \sigma * p[k, 1], x[1, 2] + \sigma * p[k, 2])) / (\sigma)^2$ ;
  end for;
  for  $k$  from 2 to 2 do
     $a[k] := c[k] / d[k]$ 
  end for;
  for  $i$  from 1 to 2 do
     $x[3, i] := x[2, i] + a[2] * p[2, i]$ 
  end for;
  for  $i$  from 1 to 2 do
     $x[1, i] := x[3, i]$ 
  end for;
  print ( $x[3, 1], x[3, 2]$ );
   $y[j, 1] := x[3, 1]$ ;  $y[j, 2] := x[3, 2]$ ;
end for

```

The asymptotic constant is calculated with great care in our computational experiments using these methods. The goal of these calculations is to show that the CGS approach outperforms the other methods considered. In the face of complicated, non-linear systems, where conventional approaches might fail or be inefficient, the CGS algorithm stands out for its efficiency and resilience.

In the following section, we will dive into the numerical results, drawing a detailed comparison between the asymptotic constants derived from the CGS method and other computational techniques.

4 Numerical results and discussion

One of the most fundamental challenges in optimization is that of developing computer algorithms for determining where the extreme points of an objective function are located. The purpose of this numerical computation is to develop an iterative technique for locating these extreme points [17].

The Rosenbrock Banana Valley function is taken into consideration as a test function for minimizing each of the methods mentioned above. We chose $(x_1, y_1) = (-1.2, 1)$ as the initial point. Each algorithm is performed in Maple with $\sigma = 0.1 \times 10^{-120}$, and accuracy is set at 400-digit, where the quotient convergent factor is calculated using the following algorithm:

Algorithm 5 Quotient Convergence factor

for i from 1 to 9 **do**

$C[i] = \text{sqrt}((y[i+1, 1] - 1)^2 + (y[i+1, 2] - 1)^2) / (\text{sqrt}((y[i, 1] - 1)^2 + (y[i, 2] - 1)^2));$

end for;

By applying the quotient convergent factor, the Rosenbrock's Banana Valley function

$$F(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

generated the following asymptotic constant shown in Table 1.

Table 1 The asymptotic constants of a nonlinear function.

Asymptotic Constant	CGS method	Newton method	Steepest Descent method	Runge-Kutta method
C_1	0.857485	0.857485	0.453589	0.785244
C_2	0.027425	0.027425	0.454544	0.785678
C_3	0.243358	0.243358	0.455502	0.786108
C_4	0.003072	0.003072	0.456463	0.786534
C_5	0.200003	0.200003	0.457428	0.786955
C_6	0.003073	0.003073	0.458396	0.787372
C_7	0.200002	0.200002	0.459367	0.787783
C_8	0.003073	0.003073	0.460341	0.788190

Table 1 shows that the final estimate for (x_0, y_0) is accurate up to six digits, but it's more than six digits. The limsup of these quotients is the quotient convergence factor, which means that the process is convergent in a quadratic fashion. The successive values of quotient for CGS method are 0.857485, 0.0027425, 0.243358, 0.003072, 0.200003, 0.003073, 0.200002, 0.003073. As a result, the quotient convergence factor oscillates and the limsup is 0.200003.

The asymptotic constants for the CGS method differ significantly from those of other methods, such the Steepest Descent method and the Runge-Kutta method. They have also reached a limit, although the limsup

value for these other approaches is far lower than 0.200003. Numerical computation is shown that other approaches for minimizing the non-quadratic function do not perform better than the CGS method [24].

Note that the asymptotic values for the CGS method and Newton's approach are practically identical. The CGS method's limit is Newton's algorithm. Because the CGS approach has no derivative, it is significantly faster than the Newton method. So a fundamental finding is that a Newton step can be substituted by a CD sequence of n linear minimizations in n well chosen directions in compared to other methods.

5 Visualization of the Rosenbrock Banana Valley function with the CGS method

In this section, we demonstrate that the CGS technique without derivatives results in the quadratic convergent and the Rosenbrock Banana Valley function reaching its minimum. This is compared to other methods, such as the Newton method or the Steepest Descent method, which are both computationally expensive and time consuming.

The Rosenbrock function, which is often commonly referred to as the "Banana Valley" function, is a non-convex function that has a particularly intricate contour shape. The fact that optimization methods, particularly those that rely on gradient descent, can easily become stuck in local minima is one of the reasons why these approaches are difficult to employ. Because of its steep cliffs and limited valley, the Rosenbrock function presented a number of severe obstacles. In order to follow the gradient without overshooting or being stuck in local minima, algorithms need to be able to adjust flexibly and efficiently.

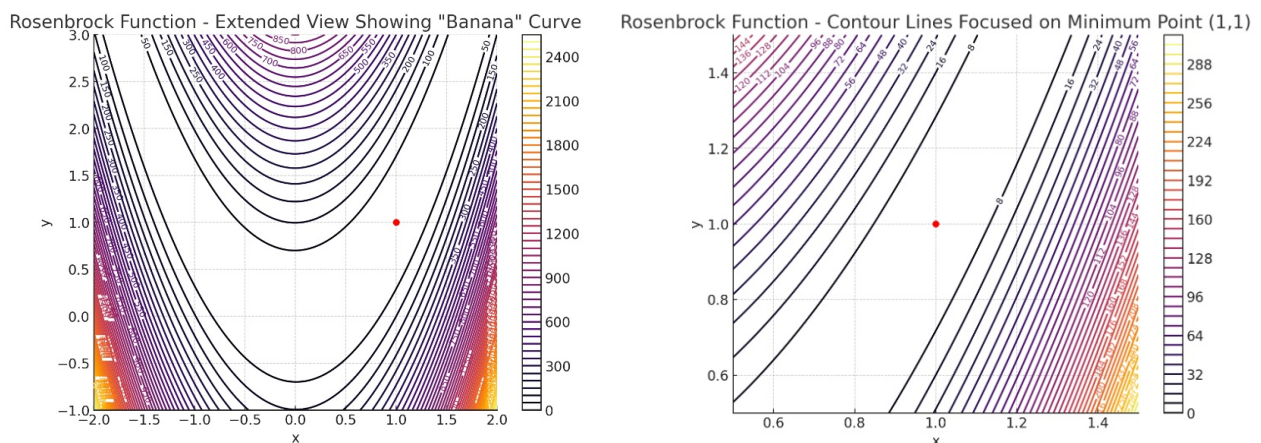


Fig. 1 Level curves of Rosenbrock's Banana Valley function.

We effectively calculate the Rosenbrock function's minimum applying conjugate gradient techniques, notably the Hestenes-Gram-Schmidt approach without derivative. For optimal navigation across the function's small valley and to guarantee orthogonality, it updates search directions based on previous gradients. Its ability to attain the minimum is further improved by its adaptive step size, resilience to beginning conditions, and non-linearities as illustrated in Figure 1.

The characteristic "banana-shaped" curve of the Rosenbrock function can be better seen in this enlarged two-dimensional plot. As they converge on the global minimum, the contour lines display the various value points of the function. Figure 1 shows the lowest point at (1,1) marked by the red dot. Based on these data, it is evident that this particular CGS without derivative technique implementation is extremely effective.

6 Discussion

The use of such techniques allows for direct numerical computations for variational problems in relation to the norms for weak and strong extremums. Such a technique has been used to solve problems in differential equations and the calculus of variations [25], in which the Banach spaces are Sobolev spaces and the second Frechet differential has a finite signature and nullity, as described by Hestenes. Other type of problems in the theory of stochastic processes. In addition to that, a variety of applications, such as radar designs by Olsen [23] and the creation of corporate feed systems for antennas and aperture distribution for antenna arrays, have benefited greatly from the use of this technique, which is very efficient in finite dimensions.

7 Conclusion

In this study, we presented a class of CD algorithms and analyzed how they performed in comparison to other effective ways of minimization. On the basis of the findings, it was determined that the CD method offered efficient minimization strategies for relatively low values of n . Both the amount of time it took for the algorithm to run and its level of complexity continuously increased as the value of n increased.

In particular, the numerical computation in Section 4 showed that the asymptotic constant changed between 0.20000 and 0.00307 for CGS-algorithm. The CGS method converged if we started anywhere in the closed convex set close to a minimum since the Hessian matrix of Rosenbrock's function was positive definite symmetric and met Sylvester's criterion. This was due to the fact that the CGS method relies on the Hessian matrix of Rosenbrock's function being positive definite symmetric in order to function.

Using quotient convergence factors, we can see from Table 1 that for Rosenbrock's function, one sequence converged quadratically. According to Ortega and Rheinboldt [14], the quotient convergence factor was almost equal to $\mathcal{Q}_2\{x^k\} = 0.200002$, which suggested quadratic convergence. The CGS algorithm often yields better results compared to other optimization methods, particularly when contrasted with Newton's method. This is because Newton's method requires the computationally intensive tasks of calculating and inverting the Hessian matrix in each iteration. It also demands the evaluation of first and second derivatives, and a line search for the right step size, which significantly increases the computational load, especially for high-dimensional problems.

8 Declarations

8.1 Conflict of interest:

There is no conflict of interests regarding the publication of this paper.

8.2 Funding:

There is no funding regarding the publication of this paper.

8.3 Author's contribution:

M.N.R.-Methodology, Analyses, Writing Original Draft, Writing Review Editing.

8.4 Acknowledgement:

The author deeply appreciate the anonymous reviewers for their helpful and constructive suggestions, which can help improve this paper further.

8.5 Data availability statement:

All data that support the findings of this study are included within the article.

8.6 Using of AI tools:

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

References

- [1] Brent R.P., Algorithms for Minimization Without Derivatives, Courier Corporation, Dover Publication, New York, USA, 2013.
- [2] Wu X., Shao H., Liu P., Zhang Y., Zhuo Y., An efficient conjugate gradient-based algorithm for unconstrained optimization and its projection extension to large-scale constrained nonlinear equations with applications in signal recovery and image denoising problems, *Journal of Computational and Applied Mathematics*, 422, 114879, 2023.
- [3] Davidon W.C., Variable metric method for minimization, Argonne National Laboratory, Lemont, Illinois, (Master Thesis), ANL-5990, University of Chicago, USA, 1959.
- [4] Fletcher R., Powell M.J.D., A rapidly convergent descent method for minimization, *The Computer Journal*, 6(2), 163–168, 1963.
- [5] Hestenes M.R., Stiefel E., Methods of conjugate gradients for solving linear systems, *Journal of Research of the National Bureau of Standards*, 49(6), 409–436, 1952.
- [6] Fletcher R., Reeves C.M., Function minimization by conjugate gradients, *The Computer Journal*, 7(2), 149–154, 1964.
- [7] Nocedal J., Wright S.J., *Numerical Optimization* (2nd Ed.), Springer, USA, 2006.
- [8] Yousif O.O.O., Mohammed M.A.Y., Saleh M.A., Elbashir M.K., A criterion for the global convergence of conjugate gradient methods under strong Wolfe line search, *Journal of King Saud University-Science*, 34(8), 102281, 2022.
- [9] Kelley C.T., *Iterative Methods for Optimization*, SIAM, Philadelphia, USA, 1999.
- [10] Zhang L., An improved Wei-Yao-Liu nonlinear conjugate gradient method for optimization computation, *Applied Mathematics and Computation*, 215(6), 2269–2274, 2009.
- [11] Doikov N., Nesterov Y., Gradient regularization of Newton method with Bregman distances, *Mathematical Programming*, 204, 1–25, 2024.
- [12] Stanimirović P.S., Shaini B.I., Sabi'u J., Shah A., Petrović M.J., Ivanov B., Cao X., Stupina A., Li S., Improved gradient descent iterations for solving systems of nonlinear equations, *Algorithms*, 16(64), 1–23, 2023.
- [13] Gupta S.D., Freund R.M., Sun X.A., Taylor A., Nonlinear conjugate gradient methods: worst-case convergence rates via computer-assisted analyses, *arXiv:2301.01530*, 2023.
- [14] Ortega J.M., Rheinboldt W.C., *Iterative solution of nonlinear equations in several variables*, SIAM, Philadelphia, USA, 1999.
- [15] Hestenes M.R., *Conjugate Direction Methods in Optimization* (Vol:12), Springer, New York, USA, 1980.
- [16] Russak I.B., Convergence of the conjugate Gram-Schmidt method, *Journal of Optimization Theory and Applications*, 33, 163–173, 1981.
- [17] Stein J.I., *Conjugate Direction Algorithms in Numerical Analysis and Optimization*, (Final Report), U.S. Army Research Office, DAHC 04-74-G-0006, National Science Foundation GP-40175, The University of Toledo, USA, 1975.
- [18] Stein J.I., Raihen M.N., Convergence rates for Hestenes' Gram-Schmidt conjugate direction method without derivatives in numerical optimization, *AppliedMath*, 3(2), 268–285, 2023.
- [19] Dennemeyer R.F., Mookini E.H., CGS algorithms for unconstrained minimization of functions, *Journal of Optimization Theory and Applications*, 16(1–2), 67–85, 1975.
- [20] Ortega J.M., Rheinboldt W.C., On a class of approximate iterative processes, *Archive for Rational Mechanics and Analysis*, 23, 352–365, 1967.
- [21] Raihen N., Convergence rates for Hestenes' Gram-Schmidt conjugate direction method without derivatives in numerical optimization, (Master Thesis), University of Toledo, USA, 2017.
- [22] Andrei N., *Nonlinear Conjugate Gradient Methods for Unconstrained Optimization*, Springer, Switzerland, 2020.
- [23] Olsen N.C., Private Communication to Ivie Stein Jr., Consultant, Lockheed, Palmdale, California, USA, 2005.
- [24] Desrochers A., Mohseni S., Quadratic optimization via conjugate directions and projection matrices, 1985 American Control Conference, IEEE, 19–21 June 1985, Boston, Massachusetts, USA, 1684–1688, 1985.
- [25] Stein I., Conjugate gradient methods in Banach spaces, *Nonlinear Analysis: Theory Methods and Applications*, 63(5–7), e2621–e2628, 2005.