A NOVEL COMBINED SUCCESSIVE GRADIENT METHOD
CONFERENCE SUBMISSIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

The search direction is crucial,common gradient-based methods typically exert a strong influence on large eigenvalues but leave small-eigenvalue components almost unchanged. If we can effectively accelerate the descent along the small-eigenvalue directions, the overall convergence speed can be significantly improved. Inspired by this, we propose a gradient scheme called CGS(Continuous Gradient Stacking). It constructs a new search vector by combining gradients from several consecutive iterations with different weights. along each eigenvalue this vector responds differently—large for small eigenvalues and small for large ones.Comparisons with BB, CBB and other standard methods demonstrate that CGS offers substantial advantages.

1 INTRODUCTION

In this paper, we present a gradient updating method for solving the unconstrained optimization problem ,the purpose is to minimize an objective function

$$minf(x) = \frac{1}{2}x^T Ax - b^T x \tag{1}$$

where $x \in \mathbb{R}^n$,$b \in \mathbb{R}^n$,$A \in \mathbb{R}^{n\times n}$ is a symmetric and positive definite matrix. The common solution methods for solving Eq.(1) are iterarive methods of the following form

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) \tag{2}$$

where α_k is a steplenth,gradient descent method and its variants are the most common optimization method.if we minimizes Eq.(1) with exact line search,then we get

$$\alpha_k = \frac{\nabla f_k^T \nabla f_k}{\nabla f_k^T A \nabla f_k} = \frac{g_k^T g_k}{g_k^T A g_k} \tag{3}$$

this method proposed by Cauchy is called steepest descent method A.Cauchy (1847),so α_k^{SD} is also called Cauchy step length. the method's convergence rate is very sensitive to ill condition number and may be very slow ,when the $f(x)$ is quadratic x_k will satisfy the

$$\frac{f(x_{k+1}) - f(x^*)}{f(x_k) - f(x^*)} \leq (\frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n})^2 \tag{4}$$

During the iteration process, the SD method exhibits a zigzag phenomena which was explained by Akaike (1959) and Forsythe (1968)

J.BARZILAI & J.M.BORWEIN (1988) presented a nonmonotone steplength which certain Quisi-Newton method, it has two choice for a_k,respectively:

$$\alpha_k^{BB1} = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}} \tag{5}$$

$$\alpha_k^{BB2} = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} \tag{6}$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$ The BB step can be seen Cauchy step with previous iteration. Barzilai and Borwein proved R-superlinear convergence rate in two dimension.Yuan (2008) for general n dimensional convex quadratic case, the method is convergent too and has a properties of R-linear rate of convergence.

there are some optimization methods based on gradient, YH (2003) decrease the gradient norm , Yuan (2006) and YH (2005) design a alternate steps , in two dimension case, it could convergence 3 steps. Serafino;F.Riccio;G.Toraldo (2013) propose SDA with a fixed stepsiz in sussesive steps. and SDC (R.De Asmundisdi SerafinoD (2014)) adding Cauchy step comparing SDA.Sun C (2020) propose new step size based on Cauchy stepsize. Z (2015) select random stepsize at some range.Raydan M (2002) propose a relax method, and introduce RSD which accelerates convergence by introducing a relaxation parameter between 0 and 2 in the standard Cauchy method, The CBB method is a nonmonotone approach that selects two directions to synthesize a new direction, they also propose CBB method which is a combination of the SD and BB method ,CBB method is essentially equivalent to the steepest descent method over two consecutive steps.

We take the reciprocal of the optimal step size, denoted by r , as our central object of study, its expression, obtained from Eq.(3), is given below. Building upon analyses of the SD and CBB methods, we then propose a continuous gradient superposition approach based on r .

$$r_k = \frac{1}{2\alpha_k} = \frac{g_k^T A g_k}{2 g_k^T g_k} \tag{7}$$

In order to make the analysis more convenient and intuitive,considering a situation the objective function is a simple n dimensions hyper-ellipsoid stimulating Eq.(1)

$$f(x) = \sum_{i=1}^{n} a^{(i)} {x^{(i)}}^2 \tag{8}$$

$$r = \frac{\sum_{i=1}^n {a^{(i)}}^3 {x^{(i)}}^2}{\sum_{i=1}^n {a^{(i)}}^2 {x^{(i)}}^2} = \frac{\sum_{i=1}^n a^{(i)} {g^{(i)}}^2}{\sum_{i=1}^n {g^{(i)}}^2} \tag{9}$$

where $0 < a^{(n)} \leq a^{(n-1)} \leq ..... \leq a^{(1)}$,$a^{(1)} \gg a^{(n)}$,$a^{(n)} < r < a^{(1)}$,$g^{(i)} = 2a^{(i)}x^{(i)}$

2 R ANYLSIS

Assuming that the current point is x_k,from Eq.(9),we have

$$r_k = \frac{\sum_{i=1}^n a^{(i)} {g_k^{(i)}}^2}{\sum_{i=1}^n {g_k^{(i)}}^2} \tag{10}$$

from Eqs.(3) (4), for SD mehthod,we have

$$x_{k+1} = x_k - \frac{\nabla f(x_k)}{2r_k} = x_k - \frac{g_k}{2r_k} \tag{11}$$

For each dimension, we have

$${x_{k+1}}^{(i)} = {x_k}^{(i)}(1 - \frac{a^{(i)}}{r_k}) = {x_k}^{(i)}\mu_k^{(i)} \tag{12}$$

$${g_{k+1}}^{(i)} = g_k^{(i)}(1 - \frac{a^{(i)}}{r_k}) = g_k^{(i)}\mu_k^{(i)} \tag{13}$$

$$r_{k+1}^{SD} = \frac{\sum_{i=1}^{n} a^{(i)} {g_{k+1}^{(i)}}^2}{\sum_{i=1}^{n} {g_{k+1}^{(i)}}^2} = \frac{\sum_{i=1}^{n} a^{(i)} {g_k^{(i)}}^2 {\mu_k^{(i)}}^2}{\sum_{i=1}^{n} {g_k^{(i)}}^2 {\mu_k^{(i)}}^2} \tag{14}$$

For CBB mehthod,its iteration is as follows Raydan M (2002),

$$x_{k+1} = x_k - 2t_k + {t_k}^2 h_k \tag{15}$$

where $h_k = Ag_k$,$t_k = \frac{g_k^t g_k}{g_k^t h_k}$.

we know that each iteration can be viewed as two consecutive steepest descent iterations in which the steplenght is computed once andused twice,Assuming that the point of the first iteration is x_k^{int},the same step size implies the same value of r,so we have

$$x_k^{int} = x_k - \frac{g_k}{2r_k} \tag{16}$$

$$x_{k+1} = x_k^{int} - \frac{g_k^{int}}{2r_k} \tag{17}$$

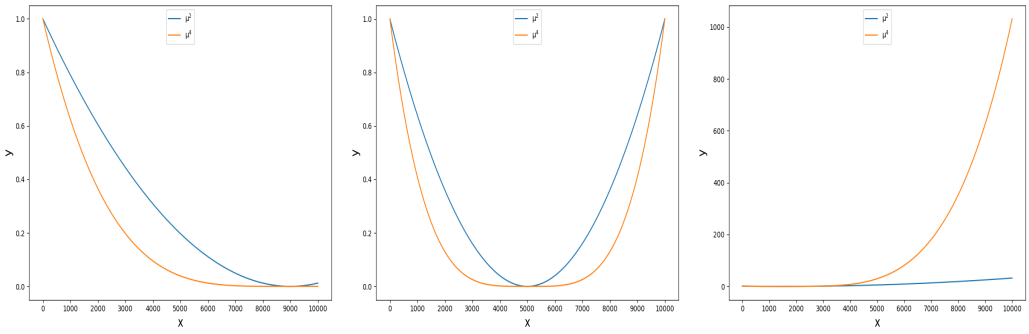
$${x_{k+1}}^{(i)} = x_k^{int(i)} - \frac{g_k^{int(i)}}{2r_k} = {x_k}^{(i)}(1 - \frac{a^{(i)}}{r_k}) - \frac{g_k^{(i)}}{2r_k}(1 - \frac{a^{(i)}}{r_k}) = {x_k}^{(i)} {\mu_k^{(i)}}^2 \tag{18}$$

$${g_{k+1}}^{(i)} = g_k^{(i)} {\mu_k^{(i)}}^2 \tag{19}$$

$$r_{k+1}^{CBB} = \frac{\sum_{i=1}^{n} a^{(i)} {g_{k+1}^{(i)}}^2}{\sum_{i=1}^{n} {g_{k+1}^{(i)}}^2} = \frac{\sum_{i=1}^{n} a^{(i)} {g_k^{(i)}}^2 {\mu_k^{(i)}}^4}{\sum_{i=1}^{n} {g_k^{(i)}}^2 {\mu_k^{(i)}}^4} \tag{20}$$

From Eqs.(12) and (18),We find that the rate of decrease for the next iteration of both the SD and CBB methods is related to the value of r or the value of μ determined by r.From Eq.(9), it can be seen that the gradient components in each dimension can be regarded as different weight values. Therefore, the value of r lies between $a^{(n)}$ and $a^{(1)}$. From Eqs.(14) and (20), We can also regard ${\mu^{(i)}}^2$ and ${\mu^{(i)}}^4$ as the weights of different gradients. We will next analyze the impact of several r values on ${\mu^{(i)}}^p$(Let p=2 or 4).As shown in Figure(1), we select three typical r values to illustrate the impact of r on μ^p.When the value of r is large, the value of μ^p is less than or equal to 1. When r is in the middle range, the value of μ^p is also less than or equal to 1 and is roughly symmetric with respect to the value of r. When r is very small, the value of μ^p shows a very large increase at higher positions.

So if the value of r is very close to $a^{(1)}$, then the value of μ lies between 0 and 1.For SD method,the rate of decrease of $a^{(i)}$ after each iteration corresponds to $\mu^{(i)}$, and after m iterations, the rate of decrease is ${\mu^{(i)}}^m$.for CBB method, the rate of decrease is ${\mu^{(i)}}^{2m}$.It can be seen that the rate of decrease for different components is the same, while the component corresponding to the smallest eigenvalue is close to 1, meaning that the smallest eigenvalue component changes very little. This implies that the overall convergence rate is very slow. If we take the different components with different orders of μ as the basis, then we can change the rate of decrease of different components by combining different weights.Specifically, it means combining components with different weights in such a way that, ultimately, the components in the direction of the small eigenvalues are increased as much as possible, while the components of the large eigenvalues are reduced.



(a) r=9000 (b) r=5000 (c) r=1500

Figure 1: $x \in [0.1, 10000]$, $\mu = 1 - \frac{x}{r}$

3 CGS METHODS

In the case of a quadratic function, we can see that the optimal point x^* is obtained when the derivative of function $f(x)$ is zero, which is equivalent to multiplying the current gradient $g(x)$ by the inverse matrix A^{-1} of A.If we consider Eq.(8), then A is a diagonal matrix with corresponding values $2a^{(i)}$, and the inverse matrix A^{-1} of A is also a diagonal matrix with corresponding values $\frac{1}{2a^{(i)}}$.If we look at it from the perspective of an input-response model, considering the current gradient as the input and the iterative process as the system, we can regard the system as a matrix. For example, for the SD method, the system matrix is a diagonal matrix with identical values, while for the Newton's method, the system matrix is A^{-1}. Therefore, the system's influence on different dimensions is different, thus producing different responses.the response for each eigenvector component can be regarded as a function that determines the rate of convergence.

The best case is to achieve the inverse of the Hessian matrix A, denoted as A^{-1}.In this way, we can reach the optimal point through a single iteration. However, we know that computing the A^{-1} requires $O(n^2)$ operations. If the dimension n is very large, then the computational cost is extremely large, making it difficult to implement. If we assume that the current point is x_k and the gradient is g_k,we know that if we iterate along the current gradient direction using a fixed step size or r value,then each iteration is equivalent to the previous component $g_k^{(i)}$ multiplied by the scaling factor $1 - \frac{a^{(i)}}{r}$.Therefore,after the k iterations, the gradient component along the i-th eigenvector is $g_k^{(i)}$ multiplied by $1 - \frac{a^{(i)}}{r}$ raised to the power of k,which is $(1 - \frac{a^{(i)}}{r})^k$.

If we treat the scaling factor as a variable σ, then during the iteration process, the factor successively becomes σ,σ^2.....σ^k and so on. It is evident that from Taylor's formula, we can see that if we combine the values of different orders of x after each iteration according to certain rules, we can obtain a response function with different response coefficients for different eigenvector components. we achieve different response coefficients for different feature components by assigning different coefficients to the gradient in each of several consecutive iterations,Thus, we obtain a response function,in the end, we speed up convergence through this function design.

Designing such a function needs to take into account the range of x, which is related to the value of r. If the value of r is close to the minimum value $a^{(n)}$, then the approximate range of x is $(-\frac{a^{(1)}}{a^{(n)}}, 0)$. If the value of r is close to the maximum eigenvalue $a^{(1)}$, then the approximate range of x is $(0, 1)$.When the condition number is large and the value of r is close to $a^{(n)}$, then each iteration is equivalent to multiplying the largest eigenvector component by the condition number $-\frac{a^{(1)}}{a^{(n)}}$. After k iterations, the largest eigenvector component is equivalent to multiplying by an extremely large quantity $(-\frac{a^{(1)}}{a^{(n)}})^k$, which may lead to the situation being uncomputable.So if we set the value of r near the maximum eigenvalue $a^{(1)}$, as analyzed earlier, it is equivalent to mapping all eigenvalues within the range from 0 to 1. In this way, the growth rate of all eigenvector components is less than 1, thus preventing any massive increase.

After obtaining the inverse matrix A^{-1} of A using Newton's method, the optimal point can be reached through a single iteration. Therefore, it would be ideal if we could simulate A^{-1}. In the form of Eq.(8), this means simulating $\frac{1}{x}$,but it is impossible to directly simulate $\frac{1}{x}$ with a polynomial, so we settle for a secondary approach, trying to make the gradient values corresponding to larger eigenvalues decrease more, and the gradient values corresponding to smaller eigenvalues decrease less more.

Additionally, we know that if we take the optimal step size along a certain vector, then taking twice the step size will lead to the same $f(x)$ value. Geometrically, from Figure(2),we can see that, in general, the direction of the eigenvector corresponding to the larger eigenvalue is closer to the boundary and will approach the boundary more quickly.

Based on the above analysis, we propose a method that combines gradients from consecutive iterations, where the gradients corresponding to different eigenvalues have different growth coefficients.If we normalize the combined gradient values, then the growth rate in the direction of the smallest eigenvalue approaches 1, while the growth rate in the direction of the largest eigenvalue approaches 0.

At the same time,we require that the function after gradient combination maintains a positive response for the gradient of each dimension, which is equivalent to multiplying the original gradient of each dimension by a positive coefficient Otherwise, if there is a negative response, the value of the corresponding component will increase with each iteration.

Assuming the eigenvalue is x and the corresponding gradient is $g(x)$,the gradient after one iteration with an iteration step size of r is $g(x) * (1 - \frac{x}{r})$. The current point is x_k, the gradient is g_k. we define $x_k = x_k^0$ and $g_k = g_k^0$.The weight coefficients are represented by a vector w that contains m values($w = [w^0, w^1.....w^m]$).then we perform m consecutive iterations with a fixed value of r.

$$x_k^{(i)} = x_k^{(i-1)}(I - A/r) = x_k^0(I - A/r)^i \tag{21}$$

$$g_k^{(i)} = g_k^{(i-1)}(I - A/r) = g_k^0(I - A/r)^i \tag{22}$$

$$G_k = \sum_{j=0}^{m} w^{(j)} g_k^j \tag{23}$$

$$x_{k+1} = x_k^i - d_{op} G_k \tag{24}$$

Where $1 \leq i \leq m$,d_{op} is the optimal step size . So a complete calculation requires first performing m iterations with a fixed value of r to obtain the combined gradient G, then making a correction to the combined gradient, and finally conducting an optimal step size calculation along the corrected combined gradient.

the value of r is slightly greater than $a^{(1)}$,the vectors in the direction of large eigenvalues are more likely to decrease.After several iterations,the components of large eigenvalues in $g_k^m$are much smaller than those in g_k.Similarly,after combination,the components of large eigenvalues in G_k may be much larger than those in g_k.This creates an imbalance between the current x_k^m value and the iteration direction G_k.Therefore, to balance the two, it is necessary to reduce the components of G_k in the direction of large eigenvalues.

We have a more intuitive explanation of the CGS algorithm on a two-dimensional plane. Since the large vector gradient decreases and the small vector gradient increases, the angle between the current point's gradient $g_k$and the combined gradient G_k obtained through gradient stacking is larger compared to the Newton gradient direction. As a result, the optimal point in the direction of G_k is closer to the optimal point x^* than that in the direction of g_k.As shown in Figure(1),If we fix the values of moving point G_k along two directions to be the same, then G_k is equivalent to rotating g_k towards the optimal point x^* by an angle.

While this method requires several calculations with fixed m step sizes for each iteration, overall it greatly reduces the computational load compared to the BB method, which necessitates an optimal step size calculation at every iteration.Compared to the CBB method, the computational load is also relatively small.We call this new Continuous Gradient Stacking(CGS) method,which is nonmonotone method.Although the CGS method is a nonmonotonic approach, its range of oscillation is small

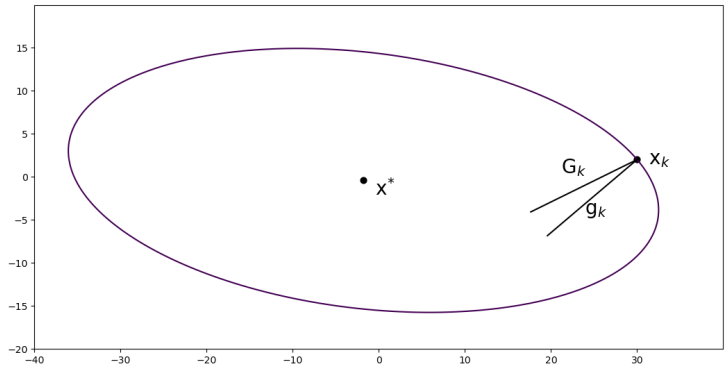


Figure 2: two-dimensional quadratic function

compared to the BB and CBB methods. The advantage of this is that it is less likely to encounter situations where the increments are too large to be calculated.

4 CGS METHOD EXAMPLE

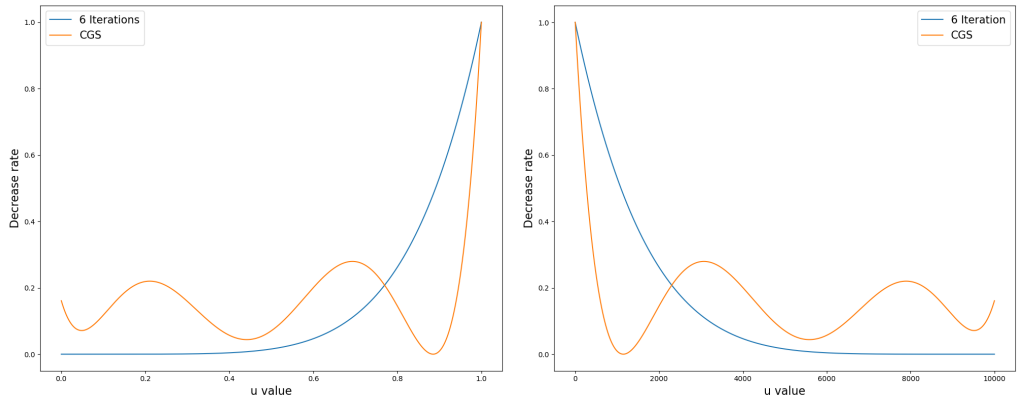
In this chapter, we set specific parameters to illustrate the CGS algorithm. If we treat $I - \frac{A}{r} > 0$ as the variable x, then $G(x) = \sum_{j=0}^{m} w^j x^j$ represents the response for each component.To facilitate analysis, we must ensure that each component of x has the same sign as the corresponding component of g. This guarantees that every component decreases when moving in the opposite direction of G_k. From Eq.(21) to Eq.(24), we set the sign of each component to remain unchanged before and after each iteration, which requires $I - \frac{A}{r} > 0$ and $G(x) > 0$. Considering an example as follow

$$f(x) = \sum_{i=1}^{100000} a^{(i)} {x^{(i)}}^2 \tag{25}$$

, where the sequence $\{a^{(i)}\}$ is arithmetic progression and $0.1 \leq a^{(i)} \leq 10000$. We set the dimension of the combined gradient to 6 and the coefficient vector w is [0.160542984,-4.328647364,65.19541352,-327.9197322,718.4158783,-710.2135166,259.6900616] so we have

$$\begin{aligned} G(x) = 0.160542984 - 4.328647364x + 65.19541352x^2 - 327.9197322x^3 \\ + 718.4158783x^4 - 710.2135166x^5 + 259.6900616x^6 \end{aligned} \tag{26}$$

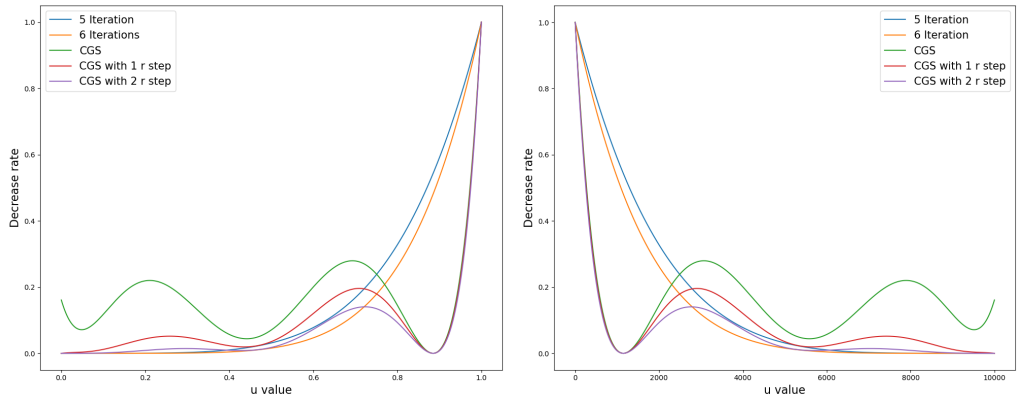
As shown in Figure3(a), the range of x is approximately from 0 to 1. If we transform it into the actual eigenvalues, the range of x would be from 0.1 to 10,000, as shown in Figure3(b).The orange line in the figure represents the CGS function. For comparison, the blue line shows the function after six consecutive iterations with r held at a fixed value. The $G(x)$ function achieves its minimum value of 4.2523e-06 at a position of approximately 0.885, corresponding to an eigenvalue of about 1150.so we can see that the combined function curve shows a sharp drop at positions less than 1150, while the overall values at positions greater than 1150 are relatively large. In particular, there are extremely high values close to or exceeding 0.2 at the positions of 3069, 7893, and 10000. If we directly optimize along the combined gradient, the large components in the direction of large eigenvalues will cause the system to quickly reach the equilibrium position, failing to effectively reduce the components corresponding to small eigenvalues.To address this issue, we need to reduce the components of the $G(x)$ function corresponding to higher eigenvalues. From Eqs.(2)(3), We know that by treating the combined gradient G_k instead of x_k as the objective to be optimized, and by taking the optimal step size, we can reduce the components corresponding to the larger eigenvalues. We call this process purification.



(a) CGS function with normalization (map eigenvalues into the interval (0,1))

(b) CGS function

Figure 3: CGS function



(a) The set of normalized functions that map eigenvalues into the interval (0,1)

(b) the set of normalized functions

Figure 4: Comparison of different functions, The blue line denotes the function $y^5 = (1 - \frac{x}{r})^5$ after five iterations;the orange line denotes the function $y^6 = (1 - \frac{x}{r})^6$ after six iterations; the green line denotes the original function CGS; the red line denotes the function $CGS^{p1} = CGS(1 - \frac{x}{r})$, which is the result of applying one purification step to CGS; and the purple line denotes the function $CGS^{p2} = CGS(1 - \frac{x}{r})^2$, obtained by applying a second purification step to CGS.

$$G_m^{pur} = G_m - \frac{\nabla f(G_m)}{r} \tag{27}$$

We can calculate $\nabla f(G_m)$ using the Hessian-Free method. Assuming that the current point is x_m, and the value of r is set slightly larger than $a^{(1)}$, we have

$$x_{m+1} = x_m - 0.5\frac{G_k}{r} \tag{28}$$

$$g_{m+1}^{(i)} - g_m^{(i)} = -\frac{a^{(i)}G_m^{(i)}}{r} \tag{29}$$

$$G_m^{(i)} + g_{m+1}^{(i)} - g_m^{(i)} = G_m^{(i)} - \frac{a^{(i)}G_m^{(i)}}{r} = G_m^{(i)}(1 - \frac{a^{(i)}}{r}) \tag{30}$$

In this way, we attenuate the large eigenvalue components of $G(x)$. As analyzed earlier,to facilitate analysis, we set the r value slightly larger than $a^{(1)}$, ensuring that in every iteration the sign of the updated gradient component matches that of the original gradient. Likewise, we require that the sign of the combined gradient also aligns with the original gradient components. For quadratic functions, the largest eigenvalue can be readily obtained using the Hessian-Free method. We know that with the Hessian-Free method, starting from the current gradient g and iteratively stepping along its direction yields Ag(A represents the Hessian matrix). stepping along Ag in turn gives A^2g. repeating this process n times gives A^ng.Because, for each dimension, this is equivalent to multiplying by ${a^{(i)}}^n$, the largest eigenvalue grows much faster than the others. Consequently, its corresponding component will eventually dominate all others. Consequently,even when n is not particularly large A^ng approximates the principal eigenvector $v_{max} = [0, 0, ..., 1]$. if at the current point x_{fr} we take a small step of size d along the direction of the largest eigenvalue to reach point x_{bk},

$$x_{fr} = x_{bk} - dv_{max} \tag{31}$$

$$g_{bk} - g_{fr} = da^{(1)} \tag{32}$$

Thus, by dividing the value obtained from Eq.(32) by d, we obtain the largest eigenvalue $a^{(1)}$ and the value of r.Following the above procedure, we obtain both the value of r and the optimized value after applying the combined gradient. In practice, the CGS algorithm faces a balance issue between the current point x (or its gradient g) and the combined gradient G. After 6 iterations, the components aligned with large eigenvalues have become very small, whereas the combined gradient still carries significant weights in those directions. Consequently, the six-iteration x(denoted as x^6) and the combined-gradient components are out of balance. Let the current gradient g_k. After normalizing both to 1,Consider the component $g_{k(3000)}$ of g_k corresponding to the eigenvalue 3000. After six iterations of gradient descent with a fixed step size r,$g_{k(3000)}^6$ is roughly 0.11. After gradient combination, the corresponding value is about 0.21, and after two CGS purification steps it falls to approximately 0.13. As shown in the Figure(4),CGS with two purification(purple line) differs markedly from y^6(orange line) but is close to y^5(blue line). To restore balance, we therefore take the x value from the fifth iteration(x_k^5) as the update, so that the two curves align more closely. Further purification of the CGS result would drive the large-eigenvalue components toward zero, yielding even better performance, at the cost of additional iterations.Therefore, from Eq.(24), our CGS formula is revised as follows

$$x_{k+1} = x_k^5 - d_{op}G_k^{p2} \tag{33}$$

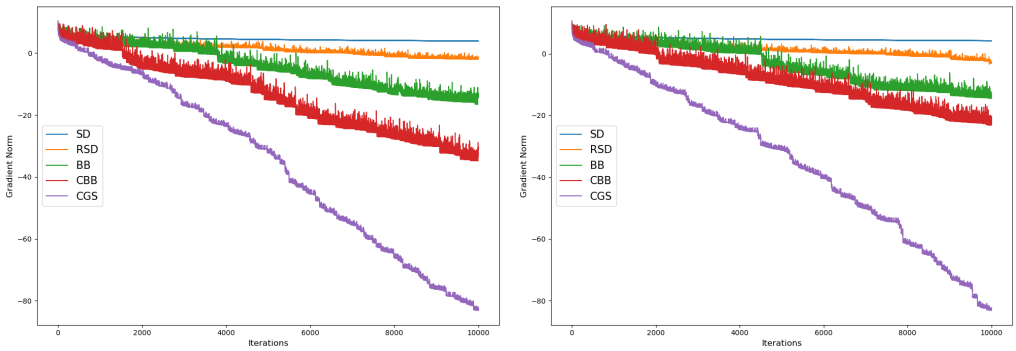
where G_k^{p2} denotes the G_k after two purification steps.

Algorithm 1 Continuous Gradient Stacking mehthod

Require: $f(x)$: objective funtion; x_0: initial point; g_0: initial gradient; d_{op}: optimal step size; ε: objective gradient norm value

Ensure: optimal x^*

initial x_0, g_0 and w
compute r value;
while $(|f(x_k|) > \varepsilon)$ **do**
  $x_k^0 = x_k^0$ and $g_k^0 = g_k$;
  $m = 0, G_k = w^{(0)} g_k^0$;
  while $(m < 6)$ **do**
    compute $x_k^{m+1} = x_k^m - \frac{g_k^m}{r}$;
    compute g_k^{m+1};
    compute $G_k = G_k + w^{(m+1)} g_k^{m+1}$;
    compute $m = m + 1$;
  end while
  compute $G_k^{p1} = G_k - \frac{Acu}{r}$;
  compute $G_k^{p2} = G_k^{p1} - \frac{AG_k^{p1}}{r}$;
  compute $x_{k+1} = x_k^5 - d_{op} G_k^{p2}$
end while



(a) random x (b) fix x

Figure 5: Comparison of 5 methods

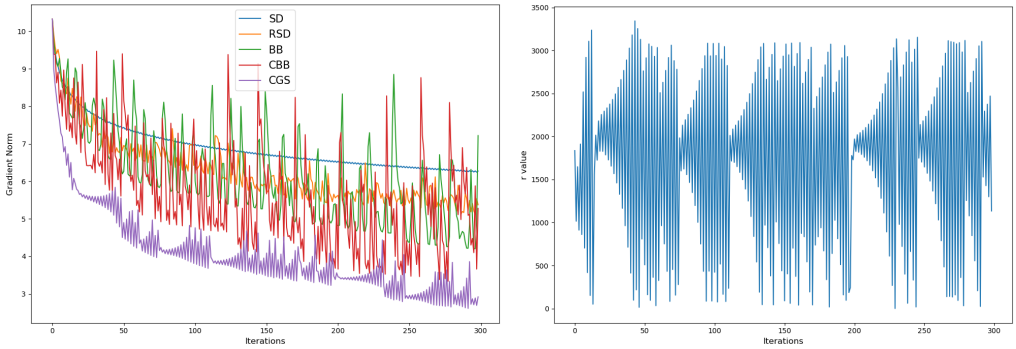
5 EXPERIMENT

We consider two groups of test problems,in the first group, the value of $\{x^{(i)}\}$ are generated according to the following furmula

$$x^{(i)} = rd * 10000 \tag{34}$$

where rd is a random number between 0 and 1,in this experiment, the values of $\{x_0^{(i)}\}$ range from 0.1764 to 9999.9238.

We compared this algorithm with the RSD method, BB method, CBB method, SD method . as shown in Figure(5a),after 10000 iterations, the minimum gradient norm values for the five methods are 6583.5315, 0.0005,2.1152e-09,8.3415e-40 and 2.7352e-75,respectively.The results showed that the CGS method is effcient for large scale problems. In second experiment we choose the initial value of $\{x_0^{(i)}\}$ to be the same, which is 10000. as shown in Figure(5b),The results are similar to the previous case,in the initial period, due to the large magnitude of each component, the CGS method will experience a significant increase in the large characteristic components after a jump, which in turn leads to a significant increase in the overall gradient norm value.As the number of iterations continues to increase, both the BB and RSD methods will exhibit a relatively slow overall decline. The SD method will hardly change. After accumulating over a period of time, the CBB method shows



(a) random 300 iterations (b) CGS r value

Figure 6: Comparison of 5 methods

a significant drop, which gives it an advantage compared to the aforementioned methods.while the CGS method will show a significant downward trend after each jump. For ease of observation, we report the results of the first 300 iterations for the five methods in Figure(6a). It can be seen that the CGS method has a significant drop compared to the other methods from the beginning. After a certain number of iterations, it stabilizes and then repeats a pattern. This pattern is that the gradient norm oscillates around a balance point with an increasing amplitude. When it reaches a very large value, the next iteration will return to a new balance and continue to repeat this process. In Figure(6b), we present the first 300 iterations of the r value for the CGS method. It can be seen that the changes in the r value are similar to the gradient norm, also showing an increasing amplitude oscillation. Additionally, it is interesting to note that, apart from the initial few iterations, the central value of r is around 1800, with larger values near 3000. As a nonmonotone gradient method, the range of fluctuations in the CGS method is much smaller compared to the BB and CBB methods. This reduces the likelihood of encountering situations where excessive changes in the direction of eigenvalues could lead to system failure. An advantage of the CGS algorithm is that, compared to CBB and BB methods, its forward-and-backward jump magnitudes are smaller, thereby avoiding system crashes caused by excessive growth increments.Another advantage of the CGS method is that it is based on several points near the current point where the changes are not significant (especially when the value of r at the current point is small). In such cases, the function is more regular, which allows for better simulation and iterative calculations.

REFERENCES

A.Cauchy. Méthode générale pour larésolution des systéms d'équations simultanées. *C.R. Acad. Sci. Paris*, 25:536–538, 1847.

H. Akaike. On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Ann. Inst. Stat. Math. Tokyo*, 11:1–17, 1959.

E. Forsythe, G. On the asymptotic directions of the s-dimensional optimum gradient method. *[j]. Numerische Mathematik*, 11:57–76, 1968.

J.BARZILAI and J.M.BORWEIN. Two point step size gradient methods. *IMA J. Numer. Anal.*, 8: 141–148, 1988.

Svaiter B F. Raydan M. Relaxed steepest descent and cauchy-barzilai-borwein method. *[J]Computational Optimization and Applications*, 21:155–167, 2002.

et al. R.De Asmundisdi SerafinoD, Hager W W. An efficient gradient method using the yuan steplength. *[J].ComputationalOptimizationandApplications*, 59(3):541–563, 2014.

R.De Asmundis;D.di Serafino;F.Riccio;G.Toraldo. On spectral properties of steepest descent methods. *IMA Journal of Numerical Analysis*, 4, 2013.

Liu JP. Sun C. New stepsizes for the gradientmethod. *[J].Optimization Letters*, 14:1943–1955, 2020.

Dai YH. Alternate step gradient method. *Optimization*, 52(4-5):395–415, 2003.

Dai YH. Analysis of monotone gradient methods. *[J]. Journal of Industrial and ManagementOptimization*, 2:181–192, 2005.

X. Yuan, Y. A new step size for the gradient method. *Journal of Computational Mathematics*, 24 (2):149–156, 2006.

X. Yuan, Y. step size for the gradient method. *Ams Ip Studies in Advanced Mathematics*, 42(2): 785–796, 2008.

Kalousek Z. Steepest descent method with random steplengths. *[J].Foundations of Computational Mathematics*, 17(2):359–422, 2015.