International Journal of Business and Management Invention (IJBMI)
ISSN (Online): 2319 – 8028, ISSN (Print): 2319 – 801X
www.ijbmi.org || Volume 13 Issue 8 || August, 2024 || PP: 33-40

A Modification of Acceleration Double Step Size algorithm for unconstrained optimization

Diptimayee Das [1] MA Adil Akthar[2] Dr. Abdul Kalam[3]
[1.] *Department of Basic Science, Einstein Academy of Technology and Management, Bhubaneswar*
[2.] *Department of Basic Science, Einstein Academy of Technology and Management, Bhubaneswar*
[3.] *Department of Basic Science, Einstein Academy of Technology and Management, Bhubaneswar*

ABSTRACT
We present a modification of acceleration double step size iteration with two vector directions. This transformation is the usage of a chosen three-term hybrid model. Derived acceleration double direction model keeps preferable properties of both include methods. Convergence analysis demonstrates at least linear convergence of the proposed iterative scheme on the set of uniformly convex and strictly convex quadratic functions. The result of numerical experiments confirms better performance profile in favor of derived hybrid acceleration double direction model when compared to its for unners.

I. Introduction and Preliminaries

The SM iteration from [1] is defined by the iterative process
y_{k+1}=y_k -$t_k\gamma_k^{-1}g_k$ (1)
Here current iterative point is y_{k+1},where$y_k$is the previous iterative point, the gradient vector g_k, t_k is a step length ,the acceleration parameter is $y_k > 0$ in [1] it is verified that the accelerated gradient SM iteration (1) out performs the gradient descent ,GD, as well as the Andrei's accelerated gradient descent AGD method from[2] . double direction and double step size accelerated methods, denoted by ADD and ADDS methods, respectively, for resolving the problems of unconstrained optimization are presentedin[3,4] thesetwomethod can be generally systematically through the next merged expression:
$y_{k+1} = y_k + \beta_k s_k + \alpha_k d_k$,(2)
$Where\ \beta_k\ and\ \alpha_k$ denote two step lengths while the vectors $s_k\ and d_k$ are two vector direction. y_k is the previous iterative point and real values . the values of the step lengthsare determined by backtracking line search techniques is basically used for defining a search direction , but some new suggestions for deriving a descending vector direction are given in [3,5] . taking the substitutions $s_k = -\gamma_k^{-1}g_k$, $\beta_K = \beta_k^2$ (3)
Into (2) produces the ADD iterative scheme from [3]:
$y_{k+1} = y_k - \beta_k\gamma_k^{-1}g_k + \beta_k^2 d_k$(4)
Where γ_k represents the acceleration parameter for the iteration (4). The parameterγ_k are explained in (3). The so called nonaccelerated version of ADD method (NADD method shortly) is defined in order to numerically verify the acceleration property of the parameter γ_k . three algorithm SM, ADD, NADD, are numerically related in [3] . derivation of the direction vector d_k is explained by the algorithm 3.2 in [3]. The ADD out performs its competitive SM method from [1] with respect to the number of iterations.
By putting the vectors s_k and d_k from (2) by $-y_k^{-1}g_k$ and $-g_k$, respectively , the next iteration is defined as
$y_{k+1} = y_k - (\beta_k\gamma_k^{-1} + \alpha_k)g_k$, (5)
The previous method is noted as ADSS model and it is $proposed\ in$ [4]. $In\ this\ article, a\ enormous$ improvement in performances of this accelerated gradient descent method when compared to the accelerated gradient descent SM method is numerically conformed the major improvement of the current paper is modification of the double step size iterative scheme (5) for unconstrained optimization into an appropriate accelerated single step size contribution is given by the numerical confirmation that the TADSS algorithm developed from the double step size ADSS model (5)is evidently more efficient than the accelerated SM method obtained in a classical way. Surprisingly numerical experiment show that the TADSSA method overcomes the initial ADSS method . the article is organized in the following way. the reduction of the double step size ADSS model into the single step size iteration TADSS and the presentation of defined accelerated gradient decent model are given in section 2. Section 3 contains the convergence
Analysis of derived algorithm for uniformly and strictly convex quadratic functions. The result of numerical experiments as well as their comparative analysis of developed method and its forerunners are illustrated in section 4.

II. Background

An idea is to investigate the properties of a single step size method developed as reduction of the double step size ADSS model this reduction is defined an additional assumption which represents a trade off between step length parameters β_k and α_k in the ADSS scheme:

$\beta_k + \alpha_k = 1$ (6)

Taking into account assumption (6) into expression 95) which defines ADSS iteration leads to the iterative process $y_{k+1} = y_k - [\alpha_k(\gamma_k^{-1} - 1) + 1]g_k$(7)

The equation (7) is modified as ADSS method , or shortly TADSS method this modification can be explained as the substitution of the product $t_k\gamma_k^{-1}$, from the SM iteration (1) by the multiplying factor $\beta_k(\gamma_k^{-1} - 1) + 1$ of the gradient from TADSS iteration (7) . for the sake of simplicity , we use the notation $\phi_k = \beta_k(\gamma_k^{-1} - 1) + 1$ whenever it is possible . the value of the acceleration parameter by using Taylor's expansion, similarly as described in [1,3,4]:

$f(y_{k+1}) \approx f(y_k) - g_k^t\ \phi_k g_k + \frac{1}{2} \in\ \phi_k g_k^t \nabla^2 f(\varepsilon)\theta_k g_k$(8)

The vector ε in (8) satisfies

$$\varepsilon\ \in [y_k, y_{k+1}], \varepsilon = y_k + \delta(y_{k+1} - y_k)$$

$= y_k - \delta\ \phi_k g_k, 0 \le \delta \le 1$(9)

Further it is reasonable to replace in (8) the Hessian $\nabla^2 f(\varepsilon)$ by the diagonal matrix $y_{k+1}\ I$,where $\gamma_{k+_1}$is an appropriately chosen real number this replacement implies

$f(y_{k+1}) \approx f(y_k) - \phi_k\left|\left|g_k\right|\right|^2 + \frac{1}{2}\ \phi_k^2\gamma_{k+1}\|\ g_k\|^2$ (10)

The relation (10) allows us to compute the acceleration parameterγ_{k+1}:

$\gamma_{k+1} = 2\frac{f(y_{k+1}) - f(y_k) + \phi_k||g_k||^2}{\phi_k^2||g_k||^2}$ (11)

Next, the natural inequality $\gamma_{k+1} > 0$ is inevitable this condition is required in order to fulfil second order necessary condition and second order sufficient condition the choice $\gamma_{k+1} = 1$ is reasonable in the case when the inequality $\gamma_{k+1} < 0$appears for some k. this choice produces the next iterative point as

$y_{k+2} = y_{k+1} - (\beta_{k+1}(1 - 1) + 1)g_{k+1} = y_{k+1} - g_{k+1}$ (12)

Which evidently represents the classical gradient decent step. We consider now the (k+1)th iteration y_{k+2} which is given by

$y_{k+2} = y_{k+1} - [\beta_{k+1}(y_{k+1}^{-1} - 1) + 1]g_{k+1} = y_{k+1} - \phi_{k+1}g_{k+1}$ (13)

Examine the function $\phi_{k+1}(\beta)$:

$\phi_{k+1}(\beta) = f(y_{k+1}) - [\beta(\gamma_{k+1}^{-1} - 1) + 1]||g_{k+1}||^2 + \frac{1}{2}[\beta(\gamma_{k+1}^{-1} - 1) + 1]^2\gamma_{k+1}\|g_{k+1}\|^2$, (14)

Defined as the finite part of the Taylor expansion of the function $f(y_{k+1} - [\beta(\gamma_{k+1}^{-1} - 1) + 1]g_{k+1})$ (15)

Under the assumption $\gamma_{k+2} = \gamma_{k+1}$.this function is convex when$\gamma_{k+1} > 0$,and its derivative $\phi_{k+1}(\beta)'_\beta$ is calculated in the following way:

$$\begin{aligned}(\phi_{k+1})'_\beta &= -(\gamma_{k+1}^{-1} - 1)\left|\left|g_{k+1}\right|\right|^2 + \left(\beta\left(\gamma_{k,+1}^{-1} - 1\right) + 1\right)\gamma_{k+1}\left|\left|g_{k+1}\right|\right|^2(\gamma_{k+1}^{-1} - 1)\\ &= (\gamma_{k+1}^{-1} - 1)(-1 + (\beta(\gamma_{k+1}^{-1} - 1) + 1)\gamma_{k+1}||g_{k+1}||^2\\ &= (\gamma_{k+1}^{-1} - 1)(-1 + \beta - \beta\gamma_{k+1} + \gamma_{k+1})||g_{k+1}||^2\\ &= (\gamma_{k+1}^{-1} - 1)(\beta - 1)(1 - \gamma_{k+1})||g_{k+1}||^2\end{aligned}$$

$= \gamma_{k+1}^{-1}(1 - \gamma_{k+1})^2(\beta - 1)||g_{k+1}||^2$ (16)

Since the inequality $\gamma_{k+1} > 0$is achieved,the following is valid:

$(\phi_{K+1})'_\beta < 0 \Leftrightarrow \beta < 1\ ,(\phi_{K+1})'_\beta = 0 \Leftrightarrow \beta = 1$ (17)

Therefore, the function $(\phi_{K+1}(\beta)$ decreases in the case $(\phi_{K+1})'_\beta < 0$ and achieves its own minimum in the case $\beta = 1$. According to the criteria given by (17), desirable values for α are within the interval $(-\infty, 1)$. Now ,(7) is a kind of the gradient decent process in the case $\phi_k = \phi\gamma_k^{-1} - 1) + 1 > 0$. Since $\gamma_k > 0$, it is easy to verify the following condition for the step length β_k:

$\beta_k \le \frac{\gamma_k}{\gamma_{k-1}}$ (18)

Since $\gamma_k/(\gamma_k - 1) > 1$ in this fractional number is not appropriate upper bound for β_k in this case. On the other hand, the inequality $\gamma_k/(\gamma_k - 1) < 0$, holds in the case $\gamma_k < 1$, so that $\gamma_k/(\gamma_k - 1)$ is an appropriate upper bound for β_k in the case.

Further analysis of (10) in the case $\gamma_k < 1$ gives

$f(y_k) - f(y_{k+1}) \approx\ \phi_k||g_k||^2 - \frac{1}{2}\ \phi_k^2\gamma_{k+1}\|g_k\|^2$ (19)

Which implies

$f(y_{k+1}) < f(y_k) \Leftrightarrow \phi_k \leq \frac{2}{\gamma_{k+1}}$ (20)

Taking into account $\gamma_k < 1$ it is not difficult to verify that the criterion (20) restricts desirable values for y_k within the interval

$\beta_k \leq \frac{\gamma_k(2-\gamma_{k+1})}{y_{k+1}(1-\gamma_k)}$ (21)

Final conclusion is that the upper bound for β_k is defined in the case$\gamma_k < 1$, $\gamma_{k+1} < 1$, as the minimum between the upper bounds defined in (18) and (21):

$\beta_k \leq \min\{\frac{\gamma_k}{\gamma_k - 1}, \frac{\gamma_k(2-\gamma_{k+1})}{\gamma_{k+1}(1-\gamma_k)}\}$ (22)

According to the previous discussion , the iterative step β_k is derived by the backtracking line search procedure presented in algorithm 1.algorithm 1 (calculation of the step size $\beta_k$by the backtracking line search which starts from the upper bound defined in (22)). Requirement: objective function f(y), the direction d_k of the search at the point y_k, and real numbers 0<$\sigma < 0.5$ and $\eta \in (\sigma, 1)$.

(1) Set $\beta = \min\{\frac{\gamma_k}{\gamma_k - 1}, \frac{\gamma_k(2-\gamma_{k+1})}{\gamma_{k+1}(1-\gamma_k)}\}$

(2)Which $f(y_k + \beta d_k) > f(y_k) + \sigma\beta g_k^t d_k take\ \beta := \eta\beta$

(3) Returnβ_k=β

Finally the TADSS algorithm of the defined accelerated gradient descent scheme (7) is presented.

Algorithm 2 (transformed accelerated double step size method (TADSS method)). Requirement: 0<$\rho < 1{,}0 < \tau < 1, y_0, \gamma_0 = 1$.

(1) Set $k = 0$, compute $f(y_0), g_0$, and take $\gamma_0 = 1$

(2)if $\|g_k\| < \varepsilon$, then go to step 8; else continue by the next step.

(3) Find the step size β_k applying algorithm 1.

(4) Computey_{k+1} using (7)

(5) Determine the scalar γ_{k+1} using (11).

(6) if $\gamma_{k+1} < 0\ or\ \gamma_{k+1} > 1$ then take $\gamma_{k+1} = 1$.

(7)Set$k := k + 1$; go to step

(8) Return$y_{k+1}$and$f(y_{k+1})$.

III. Convergence analysis

The content of this section is the convergence analysis of the TADSS method. In the first part of the section a set of uniformly convex function is considered. The proofs of the following statement can be found in [6,7] and have been omitted.

Proposition 3(see [6,7]. If the function $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ is twice continuously differentiable and uniformly convex on$\mathbb{R}^n$ then

(1)the function f has a lower bound on $l_0 = \{y \in \mathbb{R}^n | f(y) \leq f(y_0)\}$, where $y_0 \in \mathbb{R}^n$ is available;

(2) the gradient g is Lipchitz continuous is an open convex set B which contains l_0; that is, there exists $l > 0$ such that

$(\forall y, z) \in B \ \| g(y) - g(z) \| \leq l \| y - z \|$. (23)

Lemma 4. Under the assumptions of proposition 3 there exists real numbers m, M satisfying

$0 < m \leq 1 \leq M$, (24)

Such that f(y) has a unique minimisery* and

$(\forall y, z \in \ \forall y, z \in \mathbb{R}^n)\ m \| z \|^2 \leq z^t \nabla^2 f(y) z \leq m||z \|^2$,

$(\forall y, z \in \ \mathbb{R}^n) \frac{1}{2} m \| y - y^* \|^2 \leq f(y) - f(y^*) \leq \frac{1}{2} M \| \quad y - y^* \|^2$,

(25)

$(\forall y, z \in \ \mathbb{R}^n)\ m \| y - z \|^2 \leq \left(g(y) - g(z)\right)^T (y - z) \leq \quad M \| y - z \|^2$.

The value of decreasing of analysed function through each iteration is given by the next lemma which is restated and proven in [1]. The same estimation can similarly be found considering iteration (7). Theorem 6 is approved in [1] and confirms a linear convergence of the constructed method.

Lemma 5.for twice continuously differentiable and uniformly convex function f on $\mathbb{R}^n$and for the sequence $\{y_k\}$ generated by algorithm (7) the following inequality is valid:

$f(y_k) - f(y_{k+1}) \geq \mu \| g_k \|^2$, (26)

where

$\mu = \min\{\frac{\sigma}{M}, \frac{\sigma(1-\sigma)}{L}\alpha\}$. (27)

Theorem 6. If the objective function f is twice continuously differentiable as well as uniformly convex on $\mathbb{R}^n$ and the sequence $\{y_k\}$ is generated by algorithm 2 then

$\lim_{k \to \infty} \| g_k \| = 0$, (28)

And the sequence $\{y_k\}$ converges to y* at least linearly.

In the following review the case of strictly convex quadratic functions is analysed. This set of the functions is given by

$$f(y) = \frac{1}{2}y^T Ay - b^T y. \quad (29)$$

In the previous expression A is a real n×n symmetric positive definite matrix and b ∈ $\mathbb{R}^n$. It is assumed that the eigenvalue of the matrix A are given and lined as $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Since the convergence for most gradient methods is quite difficult to analyse, in many research articles of this profile convergence analysis is reduced on the set of convex quadratic [8-10]. The convergence of TADSS method is also analysed under presumptions.

Lemma 7.by applying the gradient decent method defined by (7) in which parameter γ_k and β_k are given by relation (11) and algorithm 1 on the strictly convex quadratic function f expressed by relation (29) where $A \epsilon \mathbb{R}^{n\times n}$ presents a symmetric positive definite matrix, the next inequalities hold:

$$\frac{1}{2\lambda_n} \leq \beta_{k+1}(\gamma_{k+1}^{-1} - 1) + 1 = \phi_{k+1} \leq \frac{1}{\lambda_1}, \quad (30)$$

Where λ_1 and λ_n are, respectively, the smallest and the largest eigenvalues of A.

Proof. Considering expression (29), the different between function value in the current and the previous point is

$$f(y_{k+1}) - f(y_k) = \frac{1}{2}y_{k+1}^T Ay_{k+1} - b^T y_{k+1} - \frac{1}{2}y_k^T Ay_k + b^T y_k. \quad (31)$$

Applying expression (7) the following is obtained:

$$f(y_{k+1}) - f(y_k) = \frac{1}{2}(y_k - \phi_k g_k)^T A(y_k - \phi_k g_k) - b^T(y_k - \phi_k g_k) - \frac{1}{2}y_k^T Ay_k + b^T y_k$$
$$= -\frac{1}{2}\phi_k g_k^T Ay_k - \frac{1}{2}Ay_k + \frac{1}{2}\phi_k g_k^T Ay_k + \frac{1}{2}\phi_k b^T g_k \quad (32)$$

Using the facts that the gradient of the function (29) is $g_k = Ay_k - b$ in conjunction with the equality $b^T g_k = g_k^T b$, one can verify the following:

$$f(y_{k+1}) - f(y_k)$$
$$= -\frac{1}{2}\phi_k(g_k^T Ay_k + y_k^T Ay_k - g_k^T Ag_k - 2b^T g_k)$$
$$= -\frac{1}{2}\phi_k(g_k^T(Ay_k - b) + g_k^T(Ay_k - b) - \phi_k g_k^T Ag_k)$$
$$= -\phi_k g_k^T g_k + \phi_k^2 g_k^T Ag_k. \quad (33)$$

Substituting (33) into (11) the parameter γ_{k+1} becomes

$$\gamma_{k+1} = 2\frac{-\phi_k g_k^T g_k + (1/2)\phi_k^2 g_k^T Ag_k + \phi_k g_k^T g_k}{\phi_k^2 g_k^T g_k}$$
$$= \frac{g_k^T Ag_k}{g_k^T g_k} \quad (34)$$

The last relation confirms that γ_{k+1} is the Rayleigh quotient of the real symmetric matrix A at the vectorg_k, so the next inequalities hold:

$$\lambda_1 \leq \gamma_{k+1} \leq \lambda_n, \quad k = \mathbb{N}, \quad (35)$$

Which combined with the fact $0 \leq \beta_{k+1} \leq 1$ prove the right hand side in (30):

$$\phi_{k+1} = \beta_{k+1}(\gamma_{k+1}^{-1} - 1) + 1 \leq \gamma_{k+1}^{-1} - 1 + 1 = \frac{1}{\gamma_{k+1}} \leq \frac{1}{\lambda_1} \quad (36)$$

The estimation

$$\beta_k > \frac{\alpha(1-\sigma)\gamma_k}{L}, \quad (37)$$

Proved in [1], is the considered in order to prove the left hand side of (30). Using the notation adopted in this paper, expression (37) becomes

$$\beta_k > \frac{\eta(1-\sigma)\gamma_k}{L} \quad (38)$$

Inequality (38) and the facts that $\eta \epsilon (\sigma, 1), \sigma \epsilon (0,0.5)$ and $\beta_{k+1} \epsilon (0,1)$lead to the following conclusion:

$$\phi_{k+1} = \beta_{k+1}(\gamma_{k+1}^{-1} - 1) + 1 > \frac{\beta_{k+1}}{\gamma_{k+1}} - \beta_{k+1} \geq \frac{\beta_{k+1}}{\gamma_{k+1}}\frac{\eta(1-\sigma)}{L} \geq \frac{(1-\sigma)}{L} \geq \frac{1}{2L}. \quad (39)$$

In the last estimation, Lipchitz constant L can be replaced by λ_n. The conclusion that the eigenvalue λ_n of matrix A has the property of Lipchitz constant L is to be derived from the next analysis. Since matrix A is symmetric and $g(y) = Ay - b$the following inequality can be provided:

$$\| g(y) - g(z) \| = \| Ay - Az \| = \| A(y - z) \| \leq \| A \| \left||y - z|\right| = \lambda_n \| y - z \|$$

(40)

Substituting L by λ_n, inequality (39) becomes

$$\frac{1}{2\lambda_n} \leq \beta_{k+1}(\gamma_{k+1}^{-1} - 1) + 1 \quad (41)$$

And this proves the left hand side of inequalities (30).

Remark 8. Comparing the estimations resulting from the similarly proposed lemma in [1,3,4] with the estimation derived from the previous lemma, considering the TADSS method, it can be concluded that the estimation provided by the TADSS scheme involves only the eigenvalues λ_1 and λ_n and not the parameter σ from the backtracking procedure.

Theorem 9. Let the additional assumptions $\lambda_n<2\lambda_1$ for the eigenvalue of matrix A be imposed and let f be the strictly convex function given by (29). Assume $\{v_1, v_2, \dots, v_n\}$ is the orthonormal eigenvectors of symmetric positive definite matrix A and that$\{y_k\}$ is the sequence of values constructed by algorithm 2. The gradients of convex quadratics defined by (29) are $g_k = Ay_k - b$ and can be expressed as

$g_k = \sum_{i=1}^{n} d_i^k v_i$ (42)

For some real constants $d_1^k, d_2^k, \dots d_n^k$ and for some integer K. then the application of the gradient descent method (7) on the goal function (29) satisfies the following two statement:

$(d_i^{k+1})^2 \le \delta^2 \left(d_i^k\right)^2, \delta = \max\{1 - \frac{\lambda_1}{2\lambda_n}, \frac{\lambda_n}{\lambda_1} - 1\}$(43)

$\lim_{k\to\infty} \| g_k \| = 0$ (44)

Proof. Taking into account (7) one can verify

$$g_{k+1} = A(y_k - \phi_k g_k) - b = g_k - \phi_k A g_k = (I - \phi_k A) g_k \quad (45)$$

And by taking (42) we get

$$g_{k+1} = \sum_{i=1}^{n} d_i^{k+1} v_i = \sum_{i=1}^{n} (1 - \phi_k \lambda_i) d_i^k v_i \quad (46)$$

In order to prove (43), it is enough to show that [1-$\phi_k\lambda_i| \le \delta$. So, two case have to be analysed. In the first one, it is supposed that $\lambda_i\phi_k \le 1$. Applying (30) leads to

$1 - \lambda_i\phi_k \ge \frac{\lambda_1}{2\lambda_n} \Longrightarrow 1 - \lambda_i\phi_k \le 1 - \frac{\lambda_1}{2\lambda_n} \le \delta$ (47)

In the other case, it is assumed that $\lambda_i\phi_k \ge 1$. From this condition arrives the following conclusion:

$1 - \lambda_i\phi_k \le \lambda_n \frac{1}{\lambda_1} \Longrightarrow |\lambda_i\phi_k - 1| \le \frac{\lambda_n}{\lambda_1} - 1 \le \delta.$ (48)

Expression (42) implies

$\| g_k \|^2 = \sum_{i=1}^{n} (d_i^k)^2$ (49)

The fact that the parameter δ, from (43), satisfies $0 < \delta < 1$ confirms expression (44).

IV. Numerical Result:

Numerical results provided by applying the implementation of TADSS,ADSS, SM methods on 22 test functions for unconstrained test problems, proposed in [2,11], are presented and investigated. We choose most of the functions presented in [3,4] and, as proposed in these papers, also investigated the experiments with a large number of variables in each function:1000,2000,3000,5000,

7000,8000,10000,15000,20000, and 30000. The stopping criteria are the same as in [1,3,4]. Backtracking procedure is developed using the values$\sigma = 0.0001, \eta = 0.8$ of needed parameters. Three main indicator of the efficiency are observed: number of iteration, CPU time, and number of functions evaluations. First, we compare the performance of the TADSS scheme with the ADSS method. The reasons for this section are obvious: the TADSS scheme presents aone step version of ADSS method. Also, the intention to examine behaviour of TADSS and compare it with its forerunner is natural. Obtained numerical values are displayed in table 1 and refer to the number of iterative steps, the CPU time of executions computed in seconds, and the number of evaluations of the objective function.

Obtained numerical result, generally, confirm advantages in favor of TADSS, considering all three tested indicators. More precise, regarding the number of iterative steps TADSS shows better results in 17 out of 22 functions, while ADSS outperforms TADSS in 4 out of 22 experiments and for the extended three exponential terms functions both methods require the same number of iterations. Results concerning spanned CPU time confirm that both methods, TADSS and ADSS, are very fast. In 9 out if 22 cases TADSS is faster than ADSS, in 2 out of 22 testing ADSS is faster than ADSSS, and

Table 1: summary of numerical result for TADSS and ADSS tested on 22 large scale test functions.

Test function	Number of iteration		CPU time		Number of function evaluations	
	TADSS	ADSS	TADSS	ADSS	TADSS	ADSS
Extended penalty	40	50	2	4	1280	1780
Raydan 1	466	34	11	4	8504	4844

Diagonal 1	20	37	0	0	335	1448
Diagonal 3	21	49	0	1	417	1048
Generalized tridiagonal 1	61	77	0	0	431	719
Extended tridiagonal 1	60	70	0	0	250	420
Extended three Expon. Term	40	40	0	0	400	350
Diagonal 4	40	780	0	0	270	2590
Extended Himmelblau	60	70	0	0	300	480
Quadratic QF1	4953	425	15	0	13738	1755
Extended Quad. Penalty QP1	50	60	0	2	571	841
Extended Quad. Penalty QP2	50	60	0	4	569	843
Quadratic QF2	50	60	0	1	583	836
Extended EP1	186	40	0	0	758	487
Extended tridiagonal2	638	80	0	0	2052	420
Arwhead	50	64	0	3	601	1082
Engvall	60	70	0	0	290	460
Quartic	10	70	0	0	30	390
Generalised Quartic	60	70	0	0	250	614
Diagonal 7	189	2201	0	33	566	6633
Diagonal 8	160	2213	1	40	586	6709
Diagonal 9	20	43	0	0	352	3646

Table 2. average numerical outcomes of 220 testing of each method among the 22 test functions tried out on 10 numerical experiment in each iteration..

Average performance	**TADSS**	**ADSS**
Number of iterations	331.09	302.86
CPU time (sec)	1.36	4.18
Number of function evaluation	1506.05	1745.23

even in half of examinations the CPU time of both iterations equals zero. On the issue of the number of evaluations of an objective function. TADSS improves ADSS in 17 out of 22 testing and the opposite cases. appears in 5 out of 22 table 2 displays average results of tested values.

According to results displayed in Table 2, it can be concluded that although TADSS outperforms ADSS in 17 out of 22 testing with regard to the number of iterations, average results show slight advantages of ADSS on this matter. Considering the average number of evaluations, there is an opposite case in favor of TADSS consumed CPU time is averagely three times less in favor to the TADSS comparing to the ADSS. Generally, it can be concluded that the one-step variant of the ADSS method, constructed TADSS scheme, behaves slightly better than the original

ADSS iteration, especially when we considered the speed of executions.

Some additional experiments have been carried out in further numerical research. These tests show the comparison between the TADSs and the SM iterations. As mentioned before, both of the schemes, TADSS and SM, are accelerated gradient decent methods with one iterative step size parameter. We choose this additional numerical comparison in order to confirm that the accelerated single step size TADSS algorithm, derived from the accelerated double step size ADSS model, gives better performances with respect to the all three analysed aspects than the classically defined accelerated single step size SM method. Table 3 with 30 displayed test single functions verifies the previous assertion.

It can be observed from displayed numerical outcomes that the TADSS method provides better results than the SM method considering the number of iteration in 24 out of 30 testing, while the number of opposite cases is 5 out of 30. For the NONSCOMP test functions, both models have the same number of iteration. Considering CPU time, both algorithm give the same results for 10 test functions. The TADSS method is faster than SM for 19 test functions, while the SM method is faster than TADSS for one test function only. The greatest progress is obtained with respect to the number of evaluations of the objective. On this matter, using the TADSS algorithm, better result are obtained in 27 out of 30 test functions, while the opposite case holds for two test function only. For the NONSCOMP test function both of the compared iterations give the same number of

Table 3. Numerical results for 30 test functions tested by the TADSS and the SM methods.

Test functions	Number of iteration		CPU time		Number of function evaluation	
	TADSS	SM	TADSS	SM	TADSS	SM
Extended penalty	40	589	2	5	1280	2987
Perturbed Quadratic	4276	111972	13	1868	12017	632724
Raydan 1	466	21125	11	178	8504	116757
Diagonal 1	20	10417	0	116	335	56135
Diagonal 3	21	10574	0	209	417	59425
Generalized tridiagonal 1	61	278	0	2	431	989
Extended tridiagonal 1	60	3560	0	35	250	30686
Extended three Expon. Term	40	164	0	1	400	1224
Diagonal 4	40	80	0	0	270	530
Extended Himmelblau	60	168	0	0	300	566
Quadr.diag.perturbed	11542	53133	58	1193	56359	547850
Quadratic QF1	4953	114510	15	2127	13738	649643
Extended Quad. Penalty QP1	50	224	0	12	571	2566
Extended Quad. Penalty QP2	50	162	0	7	569	2057
Quadratic QF2	50	118801	0	2544	583	662486
Extended EPI	186	68	1	1	758	764
Extended tridiagonal2	638	584	0	0	2052	2144
Arwhead	50	10	0	0	601	30
Almost perturbed quadratic	4202	110121	15	2148	14974	627287
Engvall	60	185	0	7	290	2177
Quartic	10	190	0	0	30	430
Generalised Quartic	60	156	0	0	250	423
Diagonal 7	189	90	0	0	566	220
Diagonal 8	160	96	1	0	586	594
Diagonal 9	20	11235	0	118	352	60566
DIXON3DQ	10	112899	0	1908	30	639957
NONSCOMP	10	10	0	0	30	30
HIMMELH	10	30	0	0	40	80
Power cute	1455	>t e	4	>t e	4567	>t e
index	20	>t e	0	>t e	50	>t e

Table 4: average values of numerical results for the TADSS and the SM methods calculated on 280 test for each method.

Average performance	TADSS	SM
Number of iterations	976.21	24336.82
CPU time (in seconds)	4.14	445.68
Number of evaluations	4163.68	146475.96

evaluations. From the table 3, we can also notice that for 2 out of 30 test functions testing are lasting more than the time limiter constant t_e defined in [3] while for all 30 test functions when the TADSS algorithm is applied the time of execution is far less than t_e.

The results arranged in table 4 give even more general view on the benefits provided by applying the TADSS method with regard to the SM method. To average values of 28 test functions, which were possible to test by both methods according to the constant t_e, are presented in the table.

The result presented in the previous table confirm that by applying the TADSS method approximately 25 time less iteration and even 35 times less evaluation of the objective function are needed in comparison with the SM method. Finally, when the TADSS is used, the testing is lasting even 107 times shorter than when the SM is applied.

The codes for presented numerical experiment are written in the visual C++ programming language on a workstation intel 2.2 GHz.

V. Conclusion

The accelerated single step size gradient descent algorithm, called TADSS, is defined as a transformation of the accelerated double step size gradient descent model ADSS, proposed in [4]. More precisely, the TADSS scheme is derived from the accelerated double step size gradient descent scheme ADSS by imposing relation (6) between two step parameter in the ADSS iteration. The efficiency of ADSS model regrading all analysed characteristics in comparison to the accelerated gradient descent single step size SM method has been numerically proved in [4].

The method defined in this way is comparable with its double step size forerunner, ADSS method, as well as with the single step size accelerated gradient descent SM method which is defined in a classical manner.

Results illustrated in table 1 and 2 generally indicate that the TADSS method behaves similarly as the ADSS method. From the point of mean values, the ADSS scheme gives slightly better results considering the number of iterations. On the other hand, a certain improvement regrading the number of function evaluations and needed CPU time is obtained by applying the TADSS iterations.

Even greater advantages of derived TADSS method are presented in table 3 and 4 where the comparison between the TADSS and the SM are given. Evidently, the TADSS scheme improves the SM method with respect to all three analysed characteristics, which was the prime goal in the research presented here in.

Linear convergence of the TADSS method is proved for the uniformly convex and the strictly convex quadratic functions.

Obtained results motivate further investigation of possible accelerated double step size gradient descent models and its transformations into corresponding single step size variants.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this article.

Acknowledgment

The authors gratefully acknowledge the financial support of the Serbian ministry of education, science and technological development.

References

[1]. B. Molina and M. Raydan, "Preconditioned Barzilai-BORwein method for the numerical solution of partial differential equations " Numerical Algorithms, vol 13, no. 1-2, pp. 45-60, 1996

[2]. N. Andrei,"An acceleration of gradient descent algorithm with backtracking for unconstrained optimization," Numerical Algorithm, vol. 42, no. 1, pp. 63-73, 2006.

[3]. M. J. Petrovic, "Accelerated doble step size model in unconstrained optimization," Applied mathematics and Computation, vol. 250, pp. 309-319, 2015.

[4]. M. J. Petrovic and P. S. Stanimirović, " Accelerated double direction method for solving unconstrained optimization problem," Mathematically problems in engineering, vol. 2014, Article id 965104, 8 pages, 2014.

[5]. N. I. Djuranović-Miličć and M. Gardaševic-Filipović, "A multi- step curve search algorithm in nonlinear optimization: nondifferentiable convex case," FactaUniversitatis, series: Mathematics and informatics, vol. 25,pp.11-24, 2010.

[6]. R. J. Rockafellar, convex analysis, Princeton University press, Princeton, NJ, USA, 1970.

[7]. J. M. Ortega and W. C. Rheinboldt, iterative solution of nonlin- ear Equation in several variables, Academic press, London, UK, 1970.

[8]. Y. H. Dai and L.-Z. Liao, "R-linear convergence of the Barzilai and Borwein gradient method," IMA of Numerical Analysis, vol.22, no. 1, pp. 1-10, 2002.

[9]. J. Barzilai and J. M. Browein, "Two-point step size gradient methods," IMA Journal of Numerical Analysis, vol. 8, no. 1, pp. 141-148, 1988.

[10]. B. Molina and M. Raydan, "Preconditioned Barzilai-BORwein method for the numerical solution of partial differential equations " Numerical Algorithms, vol 13, no. 1-2, pp. 45-60, 1996.