UNIVERSITY THOUGHT
Publication in Natural Sciences, Vol. 9, No. 1, 2019, pp. 57-61.

doi: 10.5937/univtho9-18174
Original Scientific Paper

COMPARATIVE PERFORMANCE ANALYSIS OF SOME ACCELERATED AND HYBRID ACCELERATED GRADIENT MODELS

MILENA PETROVIĆ[1,⋆], MILICA IVANOVIĆ[1], MARIJANA ĐORĐEVIĆ[1]

[1]Faculty of Natural Sciences and Mathematics, University of Priština, Kosovska Mitrovica, Serbia

ABSTRACT

We analyze a performance profile of several accelerated and hybrid accelerated methods. All comparative methods are at least linearly convergent and have satisfied numerical characteristics regarding tested metrics: number of iterations, CPU time and number of function evaluations. Among the chosen set of methods we numerically show which one is the most efficient and the most effective. Therewith, we derived a conclusion about what type of method is more preferable to use considering analyzed metrics.

Keywords: Gradient descent methods, Line search, Convergence rate.

ACCELERATED FACTOR IN ACCELERATED GRADIENT MODELS

We are analyzing accelerated gradient descent iterations for solving unconstrained optimization problems, mathematically described as:

$$\min f(x),\ x \in \mathbb{R}^n \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is an objective function which we want to minimize. For function f we assume that it is uniformly convex and twice continuously differentiable function. Instead of usual iterative optimization schemes, expressed by:

$$x_{k+1} = x_k + t_k d_k, \tag{2}$$

we focus on *accelerated gradient iterations* given by the following expression

$$x_{k+1} = x_k - \gamma_k^{-1} t_k d_k. \tag{3}$$

In (2) and (3) x_{k+1} stays for the next iterative function value, x_k is the current iterative function value, t_k is the iterative step size value and d_k is the search direction vector. In (3) scalar γ_k presents an iterative approximation parameter. Many authors confirmed, mostly numerically, that this parameter upgrades performance profile of posed optimization method. Nevertheless, in Stanimirović & Miladinović (2010) a class of methods containing acceleration factor is denoted as *the class of accelerated gradient methods*. From the analysis presented in Petrović & Kontrec (2017) we can conclude that one of the most efficient way for calculating acceleration parameter is through the features of the second order Taylor's series taken on the objective accelerated iteration. Although there are some alternative modes for deriving the acceleration parameter, we mention here several highly efficient accelerated models with acceleration parameter obtained by the Taylor's development: Andrei (2006, 2008); Petrović & Stanimirović (2014); Petrović (2015); Stanimirović et al. (2015); Stanimirović & Miladinović (2010). In this regard we display the expressions for acceleration parameters of some above mentioned methods:

$$\theta_k^{AGD} = -\frac{t_k g_k^T g_k}{t_k y_k^T g_k}$$

$$\gamma_{k+1}^{ADD} = 2\frac{f(x_{k+1})-f(x_k)-\alpha_k g_k^T\left(\alpha_k d_k-\gamma_k^{-1}g_k\right)}{\left(\alpha_k d_k-\gamma_k^{-1}g_k\right)^T\left(\alpha_k d_k-\gamma_k^{-1}g_k\right)}$$

$$\gamma_{k+1}^{ADSS} = 2\frac{f(x_{k+1})-f(x_k)+\left(\alpha_k\gamma_k{}^{-1}+\beta_k\right)\|g_k\|^2}{\left(\alpha_k\gamma_k{}^{-1}+\beta_k\right)^2\|g_k\|^2},$$

$$\gamma_{k+1}^{TADSS} = 2\frac{f(x_{k+1})-f(x_k)+\psi_k\|g_k\|^2}{\psi_k^2\|g_k\|^2}$$

$$\gamma_{k+1}^{SM} = 2\gamma_k\frac{\gamma_k\left[f(x_{k+1})-f(x_k)\right]+t_k\|g_k\|^2}{t_k^2\|g_k\|^2}.$$

Regarding the theory of unconstrained optimization methods, we have a unique opinion that there are two crucial elements which defined a relevant iterative optimization scheme. The first one is the vector direction, d_k, which directs the minima search. It is usually required to fulfil the descending condition:

$$g_k^T d_k < 0. \tag{4}$$

The second, equally important, is the value of the iterative step length, t_k. This element is obtained trough the exact or inexact line search procedure. In practical purpose, the inexact algorithms are certainly more preferable choice for obtaining the optimal step size iterative value. With this regard, from all above exposed, we can rightly conclude that besides these two listed elements the value of acceleration parameter, γ_k, is also important and crucial factor for one optimization method.

This paper is organized in the following way. In the second section we give an overview of some important hybrid models and hybridization process applied on accelerated gradient schemes. In the main section 3, we display obtained numerical results of four chosen models, conduct a comparative analysis and bring up a conclusion.

⋆ Corresponding author: milena.petrovic@pr.ac.rs

HYBRID MODELS

Some authors investigate a hybrid iterative systems for solving optimization problems. One of the first in this field was Picard. In his work Picard (1890) presented the following set of two iterations for solving optimization problems:

$$\begin{cases} u_1 = u \in \mathbb{C}, \\ u_{k+1} = Tu_k, \quad k \in \mathbb{N}, \end{cases} \tag{5}$$

Later on, Mann exposed his set of expressions and called it *mean value methods in iterations*

$$\begin{cases} v_1 = v \in \mathbb{C}, \\ v_{k+1} = (1-\alpha_k)v_k + \alpha_k T v_k, \quad k \in \mathbb{N}. \end{cases} \tag{6}$$

Further on, Ishikawa presented a three-term model as next:

$$\begin{cases} z_1 = z \in \mathbb{C}, \\ z_{k+1} = (1-\alpha_k)z_k + \alpha_k T y_k, \\ y_k = (1-\beta_k)z_k + \beta_k T z_k, \qquad k \in \mathbb{N}. \end{cases} \tag{7}$$

In the above displayed schemes v_k, z_k and y_k present the sequences defined by related iterations, parameters $\{\alpha_k\}, \{\beta_k\} \in (0,1)$ and $T : \mathbb{C} \to \mathbb{C}$ is a mapping defined on nonempty convex subset C of a normed space $\mathbb{E}$.

In a recent research Khan (2013), introduced the following set of relations

$$\begin{cases} x_1 = x \in \mathbb{R}, \\ x_{k+1} = Ty_k, \\ y_k = (1-\alpha_k)x_k + \alpha_k T x_k, \quad k \in \mathbb{N}. \end{cases} \tag{8}$$

In the same paper the author shows the advantages of posed process and confirms that so defined model outperforms previous three mentioned methods.

Taking good sides of the iteration set (8) the authors in Petrović et al. (2017) applied accelerated gradient descent SM method, presented in Stanimirović & Miladinović (2010), on this three-term relation. As a result a hybrid accelerated scheme is developed. We call this iteration the HSM method and it is defined by the expression:

$$x_{k+1} = x_k - \alpha t_k \gamma_k^{-1} g_k, \tag{9}$$

where parameter $\alpha \in (1,2)$ and $\gamma_k \equiv \gamma_k^{HSM}$ is iterative acceleration parameter which is computed using the second order Taylor's series of the HSM iteration

$$\gamma_{k+1} \equiv \gamma_{k+1}^{HSM} = 2\gamma_k \frac{\gamma_k \left[f(x_{k+1}) - f(x_k)\right] + (\alpha_k + 1)t_k \|g_k\|^2}{(\alpha_k+1)^2 t_k^2 \|g_k\|^2}. \tag{10}$$

In Petrović et al. (2017) the authors proved that the HSM method is at least linearly convergent on the set of uniformly convex and strictly convex quadratic functions. Numerical tests confirm significant benefits when the HSM scheme is used instead of its forerunner, the SM iteration. All these advantages indicate that this new hybridization concept can be applied on some other accelerated model and upgrade its features.

In Panić et al. (2018) some initial improvement is taken on the HSM iteration and the modified version of the HSM scheme is introduced. This model is denoted as the MHSM method. The improvement regarding the HSM iteration consists in reducing the initial step length value of the Backtracking line search algorithm. Numerical experiments show some betterment compared to the starting HSM method.

NUMERICAL COMPUTATIONS AND CONCLUSIONS DRAWN

In this section we expose comparative analysis of performance profile of four chosen methods. First comparative models is the accelerated gradient SM method presented in Stanimirović & Miladinović (2010). Second one is the hybrid accelerated method HSM from Petrović et al. (2017) which presents a hybridization of the SM method. Third model is the MHSM introduced in Panić et al. (2018) and it presents modified version of the HSM, where an initial improvement of starting value in Backtracking line search procedure was taken. Final comparative method is the accelerated gradient TADSS method which is revealed in Stanimirović et al. (2015). We now display algorithms of all listed methods:

Algorithm 0.1 SM-method Stanimirović & Miladinović (2010)

Require: Objective function $f(x)$ and chosen initial point $x_0 \in dom(f)$.

1: Set $k = 0$ and compute $f(x_0)$, $g_0 = \nabla f(x_0)$ and take $\gamma_0 = 1$.
2: If test criteria are fulfilled then stop the iteration; otherwise, go to the next step.
3: (Backtracking) Find the step size $t_k \in (0,1]$ using Backtracking procedure with $d_k = -\gamma_k^{-1} g_k$.
4: Compute $x_{k+1} = x_k - t_k \gamma_k^{-1} g_k$, $f(x_{k+1})$ and $g_{k+1} = \nabla f(x_{k+1})$.
5: Determine the scalar approximation γ_{k+1} of the Hessian of f at the point x_{k+1} using γ_{k+1}^{SM} representation.
6: If $\gamma_{k+1} < 0$, then take $\gamma_{k+1} = 1$.
7: Set $k := k+1$, go to the step 2.
8: Return x_{k+1} and $f(x_{k+1})$.

Algorithm 0.2 HSM-method Petrović et al. (2017)

Require: Function $f(x)$, $\alpha \in (1,2)$, initial point $x_0 \in dom(f)$.

1: Set $k = 0$ and calculate $f(x_0)$, $g_0 = \nabla f(x_0)$, set $\gamma_0 = 1$.
2: Check the test criteria; if stopping criteria are fulfilled then stop the algorithm; otherwise, go to the next step.
3: Applying Backtracking Algorithm: Compute the value of step size $t_k \in (0,1]$ taking $d_k = -\gamma_k^{-1} g_k$.
4: Determine $x_{k+1} = x_k - \alpha t_k \gamma_k^{-1} g_k$, $f(x_{k+1})$ and $g_{k+1} = \nabla f(x_{k+1})$.
5: Compute γ_{k+1}, approximation of the Hessian of function f at the point x_{k+1} using γ_{k+1}^{HSM} representation.
6: If $\gamma_{k+1} < 0$ take $\gamma_{k+1} = 1$.
7: $k := k+1$, go to the step 2.
8: Return x_{k+1} and $f(x_{k+1})$.

Algorithm 0.3 MHSM-method Panić et al. (2018)

Require: Function $f(x)$, $\alpha \in (1,2)$, initial point $x_0 \in dom(f)$.

1: Set $k = 0$ and calculate $f(x_0)$, $g_0 = \nabla f(x_0)$, set $\gamma_0 = 1$.
2: Check the test criteria; if stopping criteria are fulfilled then stop the algorithm; otherwise, go to the next step.
3: Applying Backtracking Algorithm: Compute the value of step size $t_k \in (0, \frac{1}{\alpha}]$ taking $d_k = -\gamma_k^{-1} g_k$.
4: Determine $x_{k+1} = x_k - \alpha t_k \gamma_k^{-1} g_k$, $f(x_{k+1})$ and $g_{k+1} = \nabla f(x_{k+1})$.
5: Compute γ_{k+1}, approximation of the Hessian of function f at the point x_{k+1} using γ_{k+1}^{HSM} representation.
6: If $\gamma_{k+1} < 0$ take $\gamma_{k+1} = 1$.
7: $k := k + 1$, go to the step 2.
8: Return x_{k+1} and $f(x_{k+1})$.

Algorithm 0.4 TADSS-method Stanimirović et al. (2015)

Require: $0 < \rho < 1, \quad 0 < \tau < 1$, x_0, $\gamma_0 = 1$.

1: Set $k = 0$, compute $f(x_0)$, g_0 and take $\gamma_0 = 1$.
2: If $\|g_k\| < \epsilon$, then go to Step 9, else continue by the next step.
3: Find the step size α_k applying Backtracking Algorithm.
4: Compute x_{k+1} using $x_{k+1} = x_k - [\alpha_k(\gamma_k^{-1} - 1) + 1]g_k$.
5: Determine the scalar γ_{k+1} using γ_{k+1}^{TADSS} representation.
6: If $\gamma_{k+1} < 0$ than take $\gamma_{k+1} = 1$.
7: Set $k := k + 1$, go to the step 2.
8: Return x_{k+1} and $f(x_{k+1})$.

We conducted numerical tests, for each comparative model, on 12 functions from Andrei (2008) for ten different number of variables: 100, 500, 1000, 1500, 2000, 3000, 5000, 7000, 8000, 10000. In the next six tables we reveal achieved results of all tested models. In order to simplify the table representations we paired the results of the SM and the MHSM algorithms and there with the results of the HSM and the TADSS. In the first three tables i.e. Tables (1) (2) and (3) we illustrate the results of obtained number of iterations, CPU time and the number of function evaluations, respectively, for the first pair of methods (SM and MHSM).

Table 1. Numerical results of *SM* and *MHSM* methods tested on 12 large scale test functions regarding number of iterations metric.

Test function	No. of iterations	
	SM	MHSM
Extended Penalty	536	449
Perturbed quadratic	41689	9228
Raydan-1	14149	7374
Extended Three Expon...	141	320
Quadratic QF1	45245	6530
Extended Quad. Penalty QP1	225	289
Extended Quad. Penalty QP2	1582	5247
Quadratic QF2	46662	11281
Extended EP1	63	217
Arwhead	228	1312
Almost Perturbed Quadratic	45098	9344
QUARTC Function	10	10

Table 2. Numerical results of *SM* and *MHSM* methods tested on 12 large scale test functions regarding CPU time metric.

Test function	No. of iterations	
	SM	MHSM
Extended Penalty	536	449
Perturbed quadratic	3	5
Raydan-1	60	80
Extended Three Expon...	0	2
Quadratic QF1	365	66
Extended Quad. Penalty QP1	0	3
Extended Quad. Penalty QP2	5	52
Quadratic QF2	504	186
Extended EP1	0	0
Arwhead	6	39
Almost Perturbed Quadratic	501	116
QUARTC Function	0	0

Table 3. Numerical results of *SM* and *MHSM* methods tested on 12 large scale test functions regarding number of function evaluations metric.

Test function	No. of func.evaluations	
	SM	MHSM
Extended Penalty	2851	5487
Perturbed quadratic	233149	74903
Raydan-1	76418	43588
Extended Three Expon...	723	1525
Quadratic QF1	253994	48796
Extended Quad. Penalty QP1	2305	2332
Extended Quad. Penalty QP2	11307	37303
Quadratic QF2	258694	100259
Extended EP1	628	1947
Arwhead	4142	13105
Almost Perturbed Quadratic	250191	76563
QUARTC Function	30	30

Table 4. Numerical results of *TADSS* and *HSM* methods tested on 12 large scale test functions regarding number of iterations metric.

Test function	No. of iterations	
	TADSS	HSM
Extended Penalty	40	400
Perturbed quadratic	11618	17086
Raydan-1	823	8377
Extended Three Expon...	40	413
Quadratic QF1	6191	15826
Extended Quad. Penalty QP1	50	338
Extended Quad. Penalty QP2	86	2704
Quadratic QF2	50	19816
Extended EP1	249	186
Arwhead	50	1023
Almost Perturbed Quadratic	11344	16980
QUARTC Function	10	10

Table 5. Numerical results of *TADSS* and *HSM* methods tested on 12 large scale test functions regarding CPU time metric.

Test function	CPU	
	TADSS	HSM
Extended Penalty	0	0
Perturbed quadratic	0	58
Raydan-1	3	32
Extended Three Expon...	0	0
Quadratic QF1	1	78
Extended Quad. Penalty QP1	0	0
Extended Quad. Penalty QP2	0	10
Quadratic QF2	0	198
Extended EP1	0	0
Arwhead	0	7
Almost Perturbed Quadratic	0	146
QUARTC Function	0	0

Table 6. Numerical results of *TADSS* and *HSM* methods tested on 12 large scale test functions regarding number of function evaluations metric.

Test function	No. of func.evaluations	
	TADSS	HSM
Extended Penalty	1123	4823
Perturbed quadratic	31349	137297
Raydan-1	10369	48952
Extended Three Expon...	400	1835
Quadratic QF1	16976	121539
Extended Quad. Penalty QP1	517	2224
Extended Quad. Penalty QP2	638	21102
Quadratic QF2	532	170950
Extended EP1	767	1514
Arwhead	549	12076
Almost Perturbed Quadratic	30838	139053
QUARTC Function	30	30

In the Tables (4), (5) and (6) we display the numerical outcomes for the second pair of methods (TADSS and HSM). Table (4) contains the number of iterations data, Table (5) contains CPU time data and Table (6) contains the number of function evaluations data, respectively, for both models.

For all tests the usual exit condition was taken:

$$\|g_k\| \leq 10^{-6} \quad \text{and} \quad \frac{|f(x_{k+1}) - f(x_k)|}{1 + |f(x_k)|} \leq 10^{-16}.$$

From last six tables we can count that the TADSS method outperform all others regarding all three analyzed metrics. Considering number of iterations in case of 8 test function TADSS is gives the best results, follows the MHSM with 2 test functions and the SM with 1 test functions. For 1 test function all methods show the same number of iterations, the same number of function evaluations and the same CPU time. Regarding the number of function evaluations, the TADSS upgrades the rest of comparative mathods in 10 test functions, while the SM gives the best results for 1 test function. Considering the CPU time metric, the TADSS shows convincingly best outcomes.

More clearer comparative view can be seen from the next Table (7) where the average values with respect to all three measured characteristics, regarding all four comparative methods, are included.

Table 7. Average numerical outcomes for 12 test functions tried out on 10 numerical experiments in each iteration.

Aver. perf.	HSM	MHSM	SM	TADSS
Num. of iter.	6929.92	4300.08	16302.33	2545.92
CPU time (sec)	44.08	52.42	148.5	0.33
Num. of fun. eval.	55116.75	33819.83	91202.67	7840.67

From the previous Table (7) we can see that the TADSS in a large degree outperforms the other three schemes regarding all three measured metrics. More precise, considering the number of iterations the TADSS shows almost three times better results than the HSM, approximately 1.7 times better than the MHSM and even 6.5 times better results than the SM method. Regrading the number of function evaluations the TADSS obtains about 7 times lower outcomes compered to the HSM method, more then 4 times lower values compared to the MHSM and about 11.5 times lower number compared to the SM method. When the CPU time is is considered, the TADSS shows nearly 133.5, 159, 0.33 and 450 times faster execution time when compared to the HSM, the MHSM and the SM respectively. This fabulous numerical outcomes in favor to the TADSS scheme are not so surprising since in Stanimirović et al. (2015) the advantage of this method in comparatione to the SM scheme is numerically confirmed. Since in Petrović et al. (2017); Panić et al. (2018) betterment of the hybrid and modified hybrid version of the SM method in comparatione to the SM iteration is detected, it is not surprising that the HSM and MHSM in Table (7) give better outcomes than the SM model.

The previous analysis lead us to an interesting question: if we make a hybrid model of the TADSS method would it be more efficient than the very TADSS iteration? The answer on this question will be revealed trough some further investigations.

From presented facts we can conclude that the hybridization process presented in Khan (2013) is a good way for improving a performance profile of the certain optimization method.

REFERENCES

Andrei, N. 2006, An acceleration of gradient descent algorithm with backtracking for unconstrained optimization. Numerical Algorithms, 42(1), pp. 63-73. doi:10.1007/s11075-006-9023-9

Andrei, N. 2008, An unconstrained optimization test functions collection. Advanced Modeling and Optimization, 10 (1), pp. 147-161.

Khan, S. 2013, A Picard-Mann hybrid iterative process. Fixed Point Theory and Applications, 2013(1), p. 69. doi:10.1186/1687-1812-2013-69

Panić, S., Petrović, M., & Carević, M. 2018, Initial Improvement Of The Hybrid Accelerated Gradient Descent Process. Bulletin of the Australian Mathematical Society, 98(02), pp. 331-338. doi:10.1017/s0004972718000552

Petrović, M. 2015, An Accelerated Double Step Size model in unconstrained optimization. Applied Mathematics and Computation, 250, pp. 309-319. doi:10.1016/j.amc.2014.10.104, 2015

Petrović, M. & Kontrec, N. 2017, Determination of accelerated factors in gradient decent iterations based on Taylor's series. University thought. Nat. Sci., 7(1), pp. 41-45. doi:10.5937/univtho7-14337

Petrović, M., Rakočević, V., Kontrec, N., Panić, S., & Ilić, D. 2017, Hybridization of accelearted gradient descent method. Numerical Algorithms, 79(3), pp 769–786, doi:10.1007/s11075-017-0460-4

Petrović, M. & Stanimirović, P. 2014, Accelerated gradient descent methods with line search. Numerical Algorithms, 54(4), pp. 503-520. doi:10.1007/s11075-009-9350-8

Picard, E. 1890, Memoire sur la theorie des equations aux derivees partielles et la methode des approximations successives. J. Math. Pures Appl., 6, pp. 145-210.

Stanimirović, P. S. & Miladinović, M. B. 2010, Accelerated gradient descent methods with line search. Numerical Algorithms, 54(4), pp. 503-520. doi:10.1007/s11075-009-9350-8

Stanimirović, P. S., Milovanović, G. V., Petrović, M., & Kontrec, N. Z. 2015, A Transformation of Accelerated Double Step Size Method for Unconstrained Optimization. Mathematical Problems in Engineering, pp. 1-8. doi:10.1155/2015/283679