Operations Research Forum (2023) 4:20
https://doi.org/10.1007/s43069-023-00193-9

ORIGINAL RESEARCH

Collinear Gradients Method for Minimizing Smooth Functions

Optimality Conditions and Algorithms

Victor K. Tolstykh[1]

Received: 11 November 2020 / Accepted: 31 December 2022

Abstract
A new optimization method for unconstrained smooth functions is proposed. It is based on a special form of the necessary optimality condition in the vicinity of the optimum. The method uses the first derivatives of the objective function, while demonstrating convergence as Newton method (second derivatives). Illustrations of how the method works step by step are provided. Algorithms for practical implementation are considered, and numerous test calculations demonstrating the high efficiency of the method are presented.

Keywords Numerical optimization · Minimization of functions · Gradient · Necessary conditions

Mathematics Subject Classification 49K10 · 49M5 · 65K05

1 Introduction

For the first time, the idea of the collinear gradients method (CollGM) was presented at the conference [1] for solving optimal control problems as an extreme problem: $u_* = \arg\min J(u)$, where u_* is a desired optimal solution (optimal control), $J(u)$ is a smooth objective function, $u \in E^n$ being a control in n-dimensional Euclidean space. To solve this problem, we will use iterative algorithms based on the gradient ∇J. The efficiency of the algorithms will be estimated by the number of calculations $J(u^k)$ and $\nabla J(u^k)$ on all iterations $k = 0, 1, 2, ...$ to achieve a given accuracy.

If the objective function is quadratic and strictly convex, then CollGM reaches the optimum u_* in one iteration, like Newton method. CollGM is a first-order method, that does not use the Hessian $H(u^k)$ with second derivatives of the objective function

✉ Victor K. Tolstykh
mail@tolstykh.com

[1] Donetsk National University, Donetsk, Russia

Published online: 14 March 2023

(as Newtonian methods do) and does not use different approximations of $H(u^k)$ (as quasi-Newtonian methods like BFGS do or methods with finite-difference representations of $H(u^k)$). CollGM, at each iteration k, makes special sub-iterations similar to the inexact Newton methods (Newton-CG type), but their sub-iterations are different.

Test calculations show that CollGM can be successfully used for non-convex optimization too. Its efficiency is no worse than different versions of Newton methods, quasi-Newton and conjugate gradient methods.

To construct CollGM algorithms, we will need a specific form of the optimality condition based on the behavior of the gradient in the vicinity of u_*. This is our starting point.

2 Optimality Conditions

In the classical theory of extreme problems, to realize the necessary optimality condition in the space E^n, the sequences of controls $u^k \xrightarrow{k\to\infty} u_*$ are considered, which in the dual space ensure the convergence to zero of the norm of the gradients:

$$||\nabla J(u^k)|| \xrightarrow{k\to\infty} 0. \tag{1}$$

In our case the dual space is E^n. Therefore, we will not further specify the space for norm and scalar product. We will consider sequences of controls that provide stronger component-based convergence of gradients, when components $\nabla_i J(u^k)$ are changed proportionally for all $i \in \{1...n\}$ from iteration to iteration. This convergence occurs when the vectors $\nabla J(u^k)$ are collinear for all k.

The essence of convergence with collinear gradients for the case of a strictly convex quadratic function $J(u)$, $u \in E^2$ is illustrated on the left in Fig. 1 on the background along the level lines (ellipses). We see that the vectors ∇J are collinear for all points of the direction d. This direction is passing through the optimum u_*.

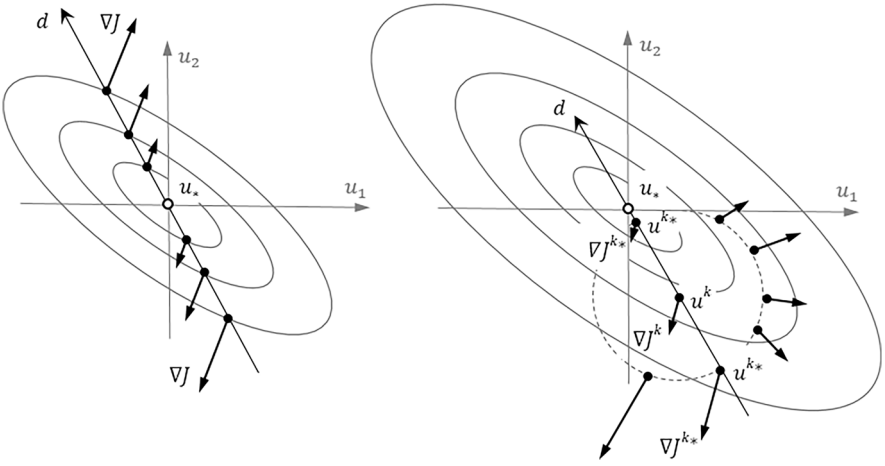

Fig. 1 The collinearity of the gradient vectors

Definition 1 The vector $\nabla J(u^k) \in E^n$ is changed *collinearly* from step k to step $k+1$ if the components of this vector are changed *proportionally*:

$$\frac{\nabla_i J(u^{k+1})}{\nabla_i J(u^k)} = const^k, \quad i \in 1 \dots n, \quad 0 < const^k < \infty, \quad k = 0, 1, 2...$$

If $\nabla_i J(u^k) = 0$ for any i, then $\nabla_i J(u^{k+1}) = 0$ is required. In this case, the ratio of zeros should be considered equal to $const^k$.

Theorem 1 *If a function $J(u)$, $u \in E^n$ is quadratic and strictly convex, then for convergence to the optimum u_*, it is necessary and sufficient that the sequence of vectors*

$$\nabla J(u^k) \xrightarrow{k \to \infty} 0 \text{ collinearly.} \tag{2}$$

Proof (Necessity) We write $J(u)$ as the following scalar product in E^n:

$$J(u) = \frac{1}{2}\langle Au, u \rangle,$$

where A is a positive definite symmetric matrix $n \times n$, $\nabla J(u) = Au$, $Au_* = 0$.

Let $d \in E^n$ be the direction from the point u^k to the point u_*. The sequence of steps $u_* - u^k \xrightarrow{k \to \infty} 0$, that lie in this direction, can be considered as a set of collinear vectors. Multiplying this collinear sequence by the matrix $-A$, we get:

$$-A(u_* - u^k) = Au^k - Au_* = \nabla J(u^k) \xrightarrow{k \to \infty} 0 \text{ collinearly.}$$

That is, for $u_* - u^k \xrightarrow{k \to \infty} 0$ in the direction d, the condition (2) must hold.

Proof (Sufficiency) The direction d is the only place to determine collinear gradients. Indeed, if the sequence $\nabla J(u^k) \xrightarrow{k \to \infty} 0$ collinearly, then multiply it with $-A^{-1}$ (the inverse matrix exists, since A is a positive definite matrix), we get:

$$-A^{-1}\nabla J(u^k) = -A^{-1}(Au^k - Au_*) = u_* - u^k \xrightarrow{k \to \infty} 0 \text{ collinearly.}$$

Therefore a sequence of collinear vectors $u_* - u^k$, directed to the point u_*, will lie on the same line d. This means that if $J(u)$ is strictly convex, quadratic, and limit (2) holds, then this is sufficient for the direction d exist and to be unique. □

If we consider that $\nabla^2 J = A \equiv H$, then the left part of the above operation can be represented as $-H^{-1}\nabla J = p_N$, where p_N is the minimization direction of Newton method, which, as is known, directed to u_* for a strictly convex quadratic function. In the right part of the above operation, we have a sequence lying in the direction d to u_*. Therefore, the following statement follows.

Corollary 1 *For a quadratic strictly convex function $J(u)$, the optimality condition (2) defines the direction d to the minimum using collinear gradients, just as Newton method defines the direction of p_N using the second derivatives*:

$$p_N = -H^{-1}\nabla J = bd,$$

where the number $b = ||H^{-1}\nabla J||/||d||$ *is a step-size.*

Classical necessary condition (1) only requires reducing the length of the vector ∇J, while ignoring the behavior of the components $\nabla_i J$, i.e., ignoring the direction of the coming to u_*. At the same time, necessary condition (2) requires approaching u_* in a strictly defined direction, namely in the direction with collinear gradients. This constitutes the fundamental differences between the necessary conditions (1) and (2).

3 Collinear Gradients Algorithms

CollGM is based on the optimality conditions of Theorem 1. In the neighborhood of any u^k we can find u^{k_*}, based on collinear gradients, and construct the direction $d^k = (u^{k_*} - u^k)$ as shown on the right in Fig. 1. Note that there can be two points u^{k_*}: the front (relative to the minimization direction, here the step-size will be $b^k > 0$) and the back (here $b^k < 0$).

If $J(u)$ is quadratic and strictly convex, then it follows from the necessary and sufficient conditions of Theorem 1, that for any initial guess u^0 there exists a direction d^0 with collinear gradients such that with an optimal step-size b^0 we reach the optimum u_* in one step (one step CollGM):

$$u_* = u^0 + b^0 d^0. \tag{3}$$

If $J(u)$ is not quadratic, then CollGM must be used at each iteration u^k, assuming that $J(u)$ in some vicinity of u^k is sufficiently close to quadratic. Here we can use the necessary condition of Theorem 1 only. Then, instead of the one-step algorithm (3), we get a multi-step CollGM:

$$u^{k+1} = u^k + b^k d^k, \qquad k = 0, 1, 2... \tag{4}$$

The step-size b^k can be easily found from the extreme condition for parabola $J(b) = J(u^k + bd^k)$. For known gradients at u^{k_*} and u^k we get:

$$b^k = \left(1 - \frac{\langle \nabla J(u^{k_*}), d^k \rangle}{\langle \nabla J(u^k), d^k \rangle}\right)^{-1}, \qquad k = 0, 1, 2... \tag{5}$$

If CollGM (4) is applied to non-convex functions, then formula (5) can lead to the maximum of $J(b)$ for some u^k. Therefore, to get the minimization step $b^k d^k$ in (4), we must use the following algorithm.

Algorithm Step for CollGM (4)

1: **procedure** GETSTEP(input $\nabla J(u^k)$, $\nabla J(u^{k_*})$, d^k; output $step$)
2: Calculate b^k from (5) and set $step = b^k d^k$.
3: **if** $\langle -\nabla J(u^k), step\rangle > 0$ **then return** $step$
else $d^k \leftarrow -step$, find $b^k = \arg\min_{b>0} J(u^k + bd^k)$, **return** $step = b^k d^k$.
4: **end procedure**

4 Choosing Collinear Gradients

In accordance with the necessary condition (2), at each step k we must find a point u in the vicinity of u^k where the gradient $\nabla J(u)$ will be collinear to $\nabla J(u^k)$. This value u is equal u^{k_*}, which we can use to do the next step $k+1$. We will evaluate the collinearity measure by the divergence (residue) of the gradient orts as [1] (for infinite-dimensional case — [2]):

$$r^k(u) = \frac{\nabla J(u)}{||\nabla J(u)||} s - \frac{\nabla J(u^k)}{||\nabla J(u^k)||}, \tag{6}$$

where if $\operatorname{sgn}\langle \nabla J(u), \nabla J(u^k)\rangle \geq 0$, then $s = 1$, else $s = -1$. The sign s allows us to estimate the collinearity of gradients both co-directional and opposite. In this case, $\max ||r^k(u)|| = \sqrt{2}$.

Expression (6) is a system of n non-linear equations whose root we must find:

$$r^k(u) = 0 \in E^n. \tag{7}$$

Problem (7) can be considered as a minimization problem for some function $F^k(u)$ whose gradient is $\nabla F^k(u) \equiv r^k(u)$. The extremum of F^k, under necessary condition $||\nabla F^k|| = 0$, gives the root of the Eq. (7), i.e., these problems are equivalent:

$$F^k(u) \to extr \quad \Longleftrightarrow \quad ||\nabla F^k(u)|| \equiv ||r^k(u)|| \to 0.$$

To solve problem (7), it is advisable to use gradient minimization methods for $F^k(u)$ since its gradient $\nabla F^k(u)$ is already known. In this case, we get iterations u^l, which is the same thing as sub-iterations u^{k_l}, and a sequence of residues $r^k(u^l) \overset{\text{def}}{=} r^l$. Usually, the Fletcher-Reeves conjugate gradient method (CGM) is used:

$$u^{k_{l+1}} = u^{k_l} + \tau^l p^l, \quad l = 1, 2, \ldots, \tag{8}$$

where $p^l = -r^l + \beta^l p^{l-1}$, $\beta^l = ||r^l||^2 / ||r^{l-1}||^2$, $\beta^1 = 0$, τ^l is a step length parameter that we will find from the assumption that the residue (6) is linear (i.e., the function $F^k(u)$ is quadratic) in a small vicinity of the point u^k. Then in this vicinity $\tau^l = ||r^l||^2 / \left\langle p^l, H^l p^l \right\rangle$ [5], where $H^l = \nabla^2 F^k(u^l)$. The vector $H^l p^l$ can be found by numerical differentiation in the direction p^l [6, 7]:

$$H^l p^l \approx \frac{r(u^{k_h}) - r(u^{k_l})}{h/||p^l||}, \tag{9}$$

where $u^{k_h} = u^{k_l} + hp^l/||p^l||$ and h is a small positive number such that $\left\langle p^l, H^l p^l \right\rangle \neq 0$.

As a criterion for successful completion of sub-iterations (8), we take the smallness of r^l relative to its maximum:

$$||r^l|| \leq c_1\sqrt{2}, \quad 0 \leq c_1 < 1,$$

where c_1 is the accuracy parameter for satisfying the collinearity of gradients.

Since we will have to make approximate calculations (assumption of local linearity of the residue, numerical differentiation, and other computational errors), we need to control the convergence of CGM (8) at each sub-iteration l.

First, we will limit the excess number of sub-iterations $l \leq l_{max}$. Besides, taking into account that l_{max} must be increased with increasing dimension n of the problem and with increasing accuracy of its solution (i.e., with decreasing c_1), we will accept

$$l_{max} = \text{integer}|c_2 \ln c_1 \ln n|, \quad c_2 \geq 1.$$

Secondly, it is advisable to control the excess number of sub-iterations under practical termination of the CGM convergence:

$$\left| \frac{||r^l|| - ||r^{l-1}||}{||r^l||} \right| \leq c_1, \quad l > 1.$$

Then, due to various noise, the direction p^l may lose its conjugacy, so it should be periodically cleared, for example, if $l = n, 2n, 3n \ldots$ then $\beta^l = 0$.

Now we need to define the first step for sub-iterations. Let it be 45 degrees angle to all coordinate axes in E^n by length δ^k in the direction of growth of $J(u^k)$:

$$u_i^{k_1} = u_i^k + \frac{\delta^k}{\sqrt{n}} \text{sgn} \nabla_i J^k, \quad i = 1 \ldots n, \quad k = 0, 1, 2..., \tag{10}$$

where δ^k is the initial radius of the vicinity of u^k. Approximately in this vicinity, we will implement necessary condition (2).

To improve the accuracy of solution (7), we will decrease the radius δ^k as the iterations u^k approach the extremum:

$$\delta^k = \min\left(\delta^{k-1} \frac{||\nabla J(u^k)||}{||\nabla J(u^{k-1})||}, \delta^0\right), \quad k = 1, 2...$$

However, there is a potential danger of "disappearing" of the descent direction when δ^k becomes too small, therefore, we should limit the minimum value of $\delta^k \geq \delta_m$, where δ_m is a small float positive number. For example, $\delta^m = 10^5 \epsilon \delta^0$, where ϵ is the machine epsilon. Moreover, the distance from u^{k_l} to u^k it is also reasonable to monitor in the same way. So, we will demand

$$\delta^k = \max\left[\min\left(\delta^{k-1}\frac{||\nabla J(u^k)||}{||\nabla J(u^{k-1})||}, \delta^0\right), \delta_m\right], \quad k = 1, 2... \tag{11}$$

and $||u^{k_l} - u^k|| \geq \delta_m, \quad l \geq 1.$

Algorithm Direction for CollGM

Parameters: $c_1, c_2, \delta^0, \delta_m, h, l_{max} = \text{integer}\,|c_2 \ln c_1 \ln n|$.
procedure GetDirection(input k, u^k, $\nabla J(u^k)$; output $\nabla J(u^{k_*})$, d^k)
  $l \leftarrow 1$. Set u^{k_l} from (10).
  If $k > 0$ **then** compute δ^k from (11).
  Compute $\nabla J(u^{k_l})$ and find r^l from (6).
  If $||r^l|| \leq c_1\sqrt{2}$ **or** $l \geq l_{max}$ **or** $||u^{k_l} - u^k|| < \delta_m$
  or $\left(l > 1 \text{ and } \left|\frac{||r^l|| - ||r^{l-1}||}{||r^l||}\right| \leq c_1\right)$ **then**
    $u^{k_*} \leftarrow u^{k_l}$, $d^k \leftarrow (u^{k_*} - u^k)$,
    return $\nabla J(u^{k_*})$, d^k, **stop**.
  If $l \neq 1, n, 2n, 3n \ldots$ **then** $\beta^l \leftarrow \frac{||r^l||^2}{||r^{l-1}||^2}$ **else** $\beta^l \leftarrow 0$.
  Compute $p^l = -r^l + \beta^l p^{l-1}$.
  Set $u^{k_h} = u^{k_l} + h\frac{p^l}{||p^l||}$, compute $\nabla J(u^{k_h})$, $r(u^{k_h})$,
  and using (9) compute $w = \langle p^l, H^l p^l\rangle$.
  If $w = 0$ **then** $h \leftarrow 10h$ and go to 7,
  else $\tau^l = \frac{||r^l||^2}{w}$ and compute $u^{k_{l+1}} = u^{k_l} + \tau^l p^l$.
  $l \leftarrow l + 1$, go to 3.
end procedure

A few words about the convergence of the obtained algorithm "Direction for CollGM". If system (8) were linear, then function $F^k(u)$ would be quadratic and its minimum would be reached in $l \leqslant n$ steps [5] (in the absence of calculation errors). But, for any quadratic and non-quadratic objective function $J(u)$, we have to minimize the non-quadratic function $F^k(u)$ by CGM with a "quadratic" step τ. This may be true in a sufficiently small vicinity of u^k. Therefore, according to (10), the step δ^k should not be large, and therefore, according to (11), δ^0 should be sufficiently small.

The parameter δ^0 should be set as the distance $||u^0 - u^{0_1}||$, at which we would like to estimate the collinearity of the gradients. For example, for a quadratic function it might be $\delta^0 \approx 0.01||u^0 - u_*||$, for a non-quadratic, δ^0 might be the distance we would like to take as the first careful step.

To minimize a quadratic function in one step by GollGM (3), do not limit the number of sub-iterations, i.e., in procedure GetDirection, the parameter c_2 should be set sufficiently large (from 2 to 5), and to obtain a satisfactory collinearity of gradients, the parameter c_1 should be set sufficiently small (from 10^{-5} to 10^{-12}). Step-size b^0 should be calculated using the formula (5).

When minimizing a non-quadratic function or quadratic function with noise by CollGM (4), it is not necessary to achieve high accuracy at each iteration k under solving (7). Here, you can take c_1 from 10^{-2} to 10^{-6} and $c_2 = 2$. The best value of c_1 depends on the type of function $J(u)$ and the initial guess u^0. Step-size b^k should be calculated using procedure GetStep.

5 Illustrations of How the Method Works for the Two-dimensional Quadratic Case

To visually illustrate the work of CollGM (3) consider the optimization of the strictly convex quadratic Schwefel 1.2 function [9] for $u \in E^2$:

$$J(u) = u_1^2 + (u_1 + u_2)^2, \quad u_* = (0, 0).$$

The results of one-step optimization by the algorithm (3), (5) with sub-iterations u^{0_l} (gray points) for the case of three starting points u^0 (black points) are shown in Fig. 2. According to (10), we performed the first step of sub-iterations $u^{0_1} - u^0$ at 45 degrees angle. For visibility, we took a fairly large $\delta^0 = 0.5$ (dotted circles). The collinearity accuracy and the number of sub-iterations were set by the parameters $c_1 = 10^4$ and $c_2 = 2$.

For all starting points u^0, there were no more than three sub-iterations, which gave us the three back points u^{0_*} and the minimization directions $-d^0 = u^{0_*} - u^0$. We can see that each CollGM minimization step $b^0 d^0 = u_* - u^0$ (where $b^0 < 0$ and satisfies (5)) corresponds to Newton steps, i.e., $b^0 d^0 = p_N$, which confirms Corollary 1.

Thus, the fulfillment of the necessary condition of the Theorem 1 (the gradients $\nabla J(u^0)$ and $\nabla J(u^{0_*})$ are collinear) implements the sufficient condition (vector $-d^0$ is directed to the optimum u_*), which allowed us to solve the problem as Newton method.

6 Tests and Illustrations for the Two-dimensional Non-convex Case

To visually illustrate the multi-step CollGM (4) consider the non-convex optimization problems [3] with $n = 2$. Here Rosenbrock function:

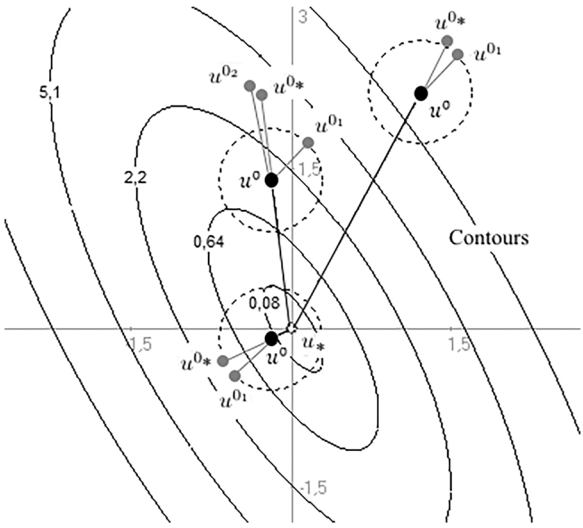

Fig. 2 Sub-iterations of the one-step CollGM

$$J(u) = 100(u_2 - u_1^2)^2 + (1 - u_1)^2, \quad u_* = (1, 1). \tag{12}$$

Himmelblau function 2:

$$J(u) = (u_2 - u_1^2)^2 + (1 - u_1)^2, \quad u_* = (1, 1). \tag{13}$$

Himmelblau function 4:

$$J(u) = 100(u_2 - u_1^3)^2 + (1 - u_1)^2, \quad u_* = (1, 1). \tag{14}$$

Himmelblau function 28:

$$\begin{aligned} J(u) &= (u_1^2 + u_2 - 11)^2 + (u_1 + u_2^2 - 7)^2, \\ u_* &\approx \{(3, 2), (-2.8, 3.1), (-3.8, -3.3), (3.6, 1.8)\}. \end{aligned} \tag{15}$$

Our cubic function:

$$J(u) = 2(u_1^3 + 2u_1^2) + (u_2^3 + 2u_2^2), \quad u_* = (0, 0). \tag{16}$$

For all functions except (15), which has four minimums, the starting point was set $u^0 = (-0.8, -1.2)$ and minimization was completed under the condition

$$||u^k - u_*||/||u^0 - u_*|| \le \epsilon = 0.01.$$

Function (15) was minimized up to $|J(u^k) - J(u^{k-1})|/J(u^0) \le \epsilon$ from the point $u^0 = (0, 0)$.

The parameters of CollGM were set to the following: c_1 from 10^{-2} to 10^{-6}; $c_2 = 2$; $\delta^0 = 0.01$, $\delta_m = 10^{-15}\delta^0$ (in our calculations $q = 20$), $h = 10^{-5}$.

Here and further, all the methods were implemented by the author in Delphi language. For comparison, the following four variants of Newton method were tested:

1. Newton — classical Newton method (analytical Hessian H) with $b^k = 1$;
2. Newton-FD — Newton method using finite-difference H via ∇J [4, 7, 8] with differentiation step $h = 10^{-8}$, $b^k = 1$;
3. Newton-TR — Trust region Newton method [4, 6, 7] with maximum (also initial) radius equal to 2;
4. Newton-CG — inexact Newton method [6, 7] with CGM sub-iterations and forcing parameter $\eta^k = \min\left(c, \sqrt{||\nabla J^k||}\right)$ [7], $c = 10^{-3}$, $b^k = 1$.

At iteration where $H \le 0$, all methods (except CollGM) were replaced by the steepest descent method (SDM) along the antigradient with the Nocedal-Wright step-size [7] under strong Wolfe condition. The step-size was calculated with the Wolfe parameters: $c_1 = 10^{-4}$; $c_2 = 0.1$; initial $\max b^k = 2\sqrt{n}$.

The optimization results are shown in Table 1, where CollGM has demonstrated good performance and better stability of calculations.

Figure 3 shows the contours of the test functions and the descent trajectories of CollGM form initial u^0 to optimum u_*. The trajectories coincide with the best executions (solid lines) of Newton method variants everywhere.

Table 1 Number of **iterations** (and calculations) for the methods

Function	Newton	Newton-FD	Newton-TR	Newton-CG	CollGM, c_1
(12)	**3** (5)	**3** (11)	**31** (98)	**3** (11)	**3** (14), 10^{-4}
(13)	**5** (9)	**5** (19)	**5** (14)	**6** (21)	**5** (18), 10^{-3}
(14)	**669** (1341)[a]	**4** (15)	**677** (2033)[a]	**6** (41)[b]	**4** (29), 10^{-6}
(15)	**8** (15)	**4** (35)[c]	**3** (8)	**4** (34)[c]	**5** (16), 10^{-2}
(16)	— [d]	**3** (19)[c]	— [d]	**4** (19)[c]	**4** (15), 10^{-2}

[a]method has done a step by SDM near zero

[b]method has done the first step by SDM

[c]method has done the first two steps by SDM

[d]all steps were done by SDM

Functions (12), (13) were minimized equally well by all methods. Function (14) turned out to be difficult for minimization by Newton and Newton-TR methods. They went along the function bottom in extremely small steps to the optimum for hundreds of iterations (see Table 1). Also, Newton-CG had not very successful trajectory (gray dotted in Fig. 2) and convergence. For function (15), the solid line in Fig. 2 shows CollGM and Newton-TR trajectories. The gray dotted line at the same figure shows Newton and Newton-CG methods trajectory, as not a good trajectory. Newton-FD for this function led to the different minimum (the lower right point). The Newton and Newton-TR methods have failed to minimize function (16) due to the analytical Hessian was always negative.

All traditional methods encountered $H < 0$ and decreased their effectiveness due to the steps done by SDM, where CollGM has never done that. Only one time CallGM in the first step for function (15) (where step is along the colinear gradients, but not along $-\nabla J$) has demanded to calculate step-size not by simple formula (5), but by the Nocedal-Wright method in procedure GetStep.

We should note that Newton-CG and CollGM are both of the first order and with CGM sub-iterations. These methods are a bit similar, but have significant differences:

1. Newton-CG method searches for direction p_N^k when $H^k p_N^k \cong -\nabla J^k$. CollGM searches for the direction d^k with collinear gradients;
2. Newton-CG method is applied at the point u^k, and CollGM is applied in the relatively large vicinity of the point u^k;
3. Newton-CG is forced to take antigradient steps when $J(u)$ is non-positively convex. CollGM takes steps only in the direction of collinear gradients for any convexity;
4. CollGM very rarely uses to the choice of a step-size b^k, which requires additional expensive calculations other than the simple formula (5).

Since CollGM and Newton-CG are formally close to each other in terms of implementation technique, we will further test their capabilities at different initial gueses of u^0 on the function (14). The results are presented in Table 2. As we can see, Newton-CG method in most cases was unable to minimize the function. Often $H < 0$ appeared

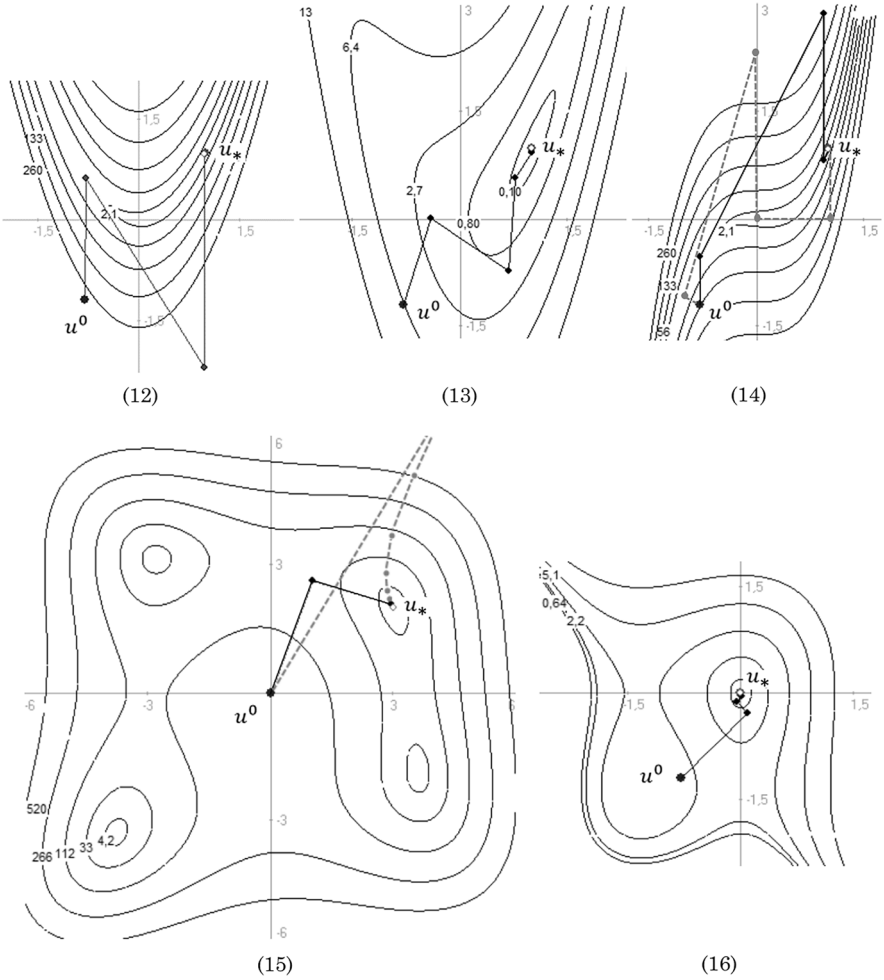

Fig. 3 Minimizing by the collinear gradients

near the bottom, and the following antigradient steps took the solution far from the optimum (the step-size was almost infinite). Apparently, Newton-CG with trust region method should have been used here. In contrast, CollGM demonstrated high performance and good efficiency at all u^0. The latter never had $H < 0$ and performed all the steps by a simple formula (5).

Table 2 Number of **iterations** (and calculations) for the function (14)

Method	Initial guess u^0					
	$(2, -0.8)$	$(0.2, 2)$	$(-0.8, 1, 4)$	$(0, 0)$	$(-1.2, 0, 2)$	$(0.2, -1, 2)$
Newton-CG	—	—	—	**2** (6)	—	**3** (11)
CollGM	**4** (23)	**3** (18)	**4** (41)	**2** (11)	**4** (33)	**3** (20)

7 Tests for the Multidimensional Non-convex Case

Consider the generalized Rosenbrock function [10, 11], a variant that is considered the most difficult to analyze, and numerical minimization [12]:

$$J(u) = \sum_{i}^{n-1} \left[100(u_i^2 - u_{i+1})^2 + (u_i - 1)^2\right]. \tag{17}$$

In [12] for the case $n = 30$, it is shown that in the area $U = \{u_i \in [-1, 1]\}_i^n$, function (17) has 108 stationary points, including the global minimum $u_* = (1, \dots, 1)$, and local minimum $u \approx (-1.1, \dots, 1)$. The other 106 are saddle points. Optimization [12] was implemented by running Newton method 10^8 times with a random u^0 from a given U.

We also took $n = 30$ and set of eight various u^0. The test conditions for CollGM were the same as in the previous section with the exception $\delta^0 = 0.1$ (where $\delta^0 \approx 0.02 \max_U ||u||_{E^{30}}$). For comparison with CollGM, the traditional methods of the first order were tested: Fletcher-Reeves and Polak-Ribière CGM [5, 8] with restart at every 3n iterations, quasi-Newton BFGS [3, 4, 6, 7] and LBFGS [6, 7] with a memory of the last five steps and Newton-CG. All methods (except Newton-CG where $b^k = 1$) were used with a step-size b^k according to the golden section method (up to $||u^{k+1} - u^k|| < \epsilon||u^0 - u_*||$) and Nocedal-Wright method. In total, there were nine algorithms plus CollGM.

All algorithms using the Nocedal-Wright method for b^k ended up with convergence far from the u_* with $b^k \cong 0$. The real results came with b^k by the golden section method. Under these conditions, the Fletcher-Reeves algorithm was much better than the Polak-Ribière, and LBFGS was almost always was more efficient than BFGS. Newton-CG was able to minimize the function in only three cases of u^0 out of eight.

Testing results of Fletcher-Reeves CGM, LBFGS (both with golden section) and CollGM are shown in Table 3. CollGM has demonstrated high performance, and the best efficiency. The step parameter b^k was almost always calculated by a simple formula (5), in rare cases the Nocedal-Wright step was used, namely, when u^0 was equal to: 3 (two steps); 4, 5 (one step); 7 (four steps). Note that CollGM under these conditions showed approximately the same results using the golden section as when using the Nocedal-Wright.

Table 3 Number of **iterations** (and calculations) for different initial guesses

Initial guess u^0		CGM	LBFGS	CollGM, c_1
1	$(-1.2, 1, \dots, -1.2, 1)$	**602** (6030)	**385** (3860)	**76** (771), 10^{-3}
2	$(-0.8, -1.2, \dots, -0.8, -1.2)$	**604** (5445)	**272** (2457)	**48** (951), 10^{-4}
3	$(0.5, -1.2, \dots, 0.5, -1.2)$	**500** (5010)	**386** (3870)	**54** (722), 10^{-3}
4	$(-1, \dots, -1)$	**685** (6174)	**267** (2412)	**49** (976), 10^{-4}
5	$(1.2, -1.2, \dots, 1.2, -1.2)$	**695** (6960)	**392** (3930)	**73** (766), 10^{-3}
6	$(2, 0.8, \dots, 2, 0.8)$	**62** (687)	**2432** (26757)	**13** (392), 10^{-4}
7	$(-1, \dots, 1)$	**719** (7199)	**898** (8989)	**56** (935), 10^{-5}
8	$(-1, \dots, 0)$	**625** (6256)	**385** (3856)	**49** (1072), 10^{-4}

Recall that the parameter c_1 sets the gradient collinearity accuracy for the quadratic approximation of $J(u)$ at the point u^k. Obviously, when minimizing a non-quadratic function, it is hardly possible to preset the best value of c_1 for all u^k. Here a numerical experiment is necessary, which was done for Table 3.

8 Conclusions

Based on the necessary condition for optimality of Theorem 1, we were able to construct collinear gradients algorithms and achieve good convergence to the optimum in a convex quadratic problem and in a set of complex non-convex problems. CollGM has demonstrated results as good as, and in some cases better than various variants of Newton method, quasi-Newton methods, conjugate gradient methods. When minimizing quadratic functions, CollGM behaves completely like Newton method, which is explained by Corollary 1.

The presence of configurable parameters of the method (primarily c_1, as well as c_2, δ^0) on the one hand makes its use informal, but on the other hand allows us to configure the method to effectively solve complex problems where other methods are extremely inefficient or even unusable.

Data Availability Data sharing not applicable to this article as no data sets were generated or analyzed during the current study.

Declarations

Conflict of Interest The author declares is no competing interests.

References

1. Tolstykh VK (2000) New first-order algorithms for optimal control under large and infinite-dimensional objective functions. In: Deville M & Owens R (eds) Proc. of the 16th IMACS World Congress on Sc. Computation, Appl. Mathematics and Simulation, Lausanne-Switzerland, pp 307
2. Tolstykh VK (2012) Optimality conditions and algorithms for direct optimizing the partial differential equations. Engineering 7:390–393
3. Himmelblau DM (1972) Applied Nonlinear Programming. McGraw-Hill, New York
4. Dennis JE, Schnabel Robert B (1996) Numerical methods for Unconstrained Optimization and Nonlinear Equations. SIAM, NJ
5. Fletcher R (1987) Practical Methods of Optimization. John Wiley & Sons, New York
6. Kelley CT (1999) Iterative Methods for Optimization. SIAM, Philadelphia
7. Nocedal J, Wright SJ (2000) Numerical Optimization. Springer, New York
8. Polak E (1997) Optimization: algorithms and consistent approximations. Springer, New York
9. Schwefel H-P (1995) Evolution and Optimum Seeking. Wiley Interscience, New York
10. Andrei Neculai (2008) An Unconstrained optimization test functions. Advanced Modeling and Optimization 1:147–161
11. Shan Y-W, Qiu Y-H (2006) A note on the extended Rosenbrock function. Evol Comput 1:119–126
12. Kok S, Sandrock C (2009) Locating and characterizing the stationary points of the extended rosenbrock function. Evol Comput 3:437–453