New first-order algorithms for optimal control under large and infinite-dimensional objective functions

Victor K. Tolstykh *

March 24, 2000

Abstract

The complicated systems under large-dimensional controls and controls as functions can usually not be optimized by the traditional computing first-order methods. The second-order methods can be implemented not always. There is proposed a peculiar interpretation of a necessary condition for optimality for a convex unconstrained objective function. The condition is written for a descent trajectory and not just for a stationary point. It is realized by the simple optimization first-order methods. There are comparative computing tests for large and infinite-dimensional controls (parabolic and hyperbolic equations). The new methods show a high efficiency. There are Internet URLs at the test software.

Key words: necessary condition for optimality, optimization, gradient.

AMS subject classifications: 49K10, 49M40.

1 Introduction

Given the objective function $J(u)$ with a unique local minimum at u_*. We need to find the optimal control

$$u_* = \arg \min_{u \in R^n} J(u), \tag{1}$$

where R^n denotes a real n-dimensional Euclidean space and n is a large number or $n = \infty$ if control u is a function. In the last case $u \in L_2$ usually. The objective function J can depend from a control u very complicated. Such dependence can include differential equations. We are going to solve the nonlinear problem (1) by iterations:

$$u^{k+1} = u^k + b^k d(u^k), \qquad b^k > 0, \quad k = 0, 1, \ldots \tag{2}$$

where $d(u^k) \equiv d^k$ is a descending direction of objective J at point u^k and number b^k is a step-size along direction d^k. Usually direction d^k is the anti-gradient $-\nabla J$. To calculate the gradient we can use a Lagrange multiplier technique. Direct minimization of the objective function is imagined the most simple and universal approach to solve the nonlinear optimal control problems. However not all so is simple as it seems.

All optimization methods differ from each other by direction d^k and by number b^k. Let us briefly describe advantages and disadvantages of the traditional optimization methods.

*Computer Scince Department, Donetsk State University, Universitetskaya-24, 83055 Donetsk, Ukraine (tvk@dongu.donetsk.ua, http:// www.dongu.donetsk.ua/fizfak/kkt2/tol.htm)

1.1 Steepest descent method (SDM)

In algorithm (2) we have $d^k = -\nabla J^k$ and a step-size b^k from , e.g.,

$$b^k = \arg\min_{b \geq 0} J(b), \quad \text{where } J(b) = J(u^k + bd^k). \tag{3}$$

There are known theorems [8],[15] which guarantee achievement of $\min J$ for any initial guess u^0 only for $k \to \infty$ even for case of 2-dimensional quadratic J. This fact is that SDM is impossible to employ on a large-scale minimization without a successful initial guess u^0. How to select a successful u^0 nobody knows.

1.2 Conjugate gradient methods (CGM)

Here minimization direction d^k is calculated from a conjugacy condition relative to a previous direction d^{k-1} [5],[10] and others. For example, Fletcher–Reeves' direction $d^k = -\nabla J^k + \left\|\nabla J^k\right\|^2 / \left\|\nabla J^{k-1}\right\|^2 d^{k-1}$, and a step-size b^k is computed by (3).

Those methods should minimize the convex quadratic function $J(u)$ in, at most, n iterations (if computing errors are absent) for anyone initial guess u^0 [8]. This circumstance is positive, however, CGM is very sensitive to computing errors, therefore for large n the effectiveness of CGM can sharply drop.

1.3 Newton method (NM)

The Newton direction is $d^k = -(\nabla^2 J^k)^{-1} \nabla J^k$, where $\nabla^2 J$ is a second derivative of $J(u)$. The NM is the second-order one. This method minimizes the convex quadratic function in one iteration for $b^k = 1$ for any initial guess u^0. The positive index is obvious. Now we shall consider the negative index. The second derivative represents a Hesse $n \times n$ matrix H. For large n calculating the second partial derivatives in a matrix H (and then inverse: H^{-1}) it can reduce into a non-solvable task.

If representation of H is made by finite differences including ∇J, then we will have discrete NM [4],[7],[8], which is a first-order method. This representation for any iteration k requires $n+1$ times of computing the gradients and the inverting a received approximate Hessian. That is very ineffective for large n and not enough studied for infinite n.

1.4 Quasi-Newton methods (QNM)

If representation of a Hessian H (or H^{-1}) is made by iteration progress on the basis of gradient ∇J, then we will have QNM [3],[4],[6], [8],[11]. They are first-order methods but they use very complicated algorithms and large storage $n \times n$ for matrixes. For quadratic J QNM approximate H^{-1} in n iterations and can be more effective than CGM. Generalizations of QNM on infinite-dimensional problems are absent.

We see the traditional optimization methods interfere with the large difficulties for large n and have not the grounding under $n = \infty$. We are going to consider new first-order algorithms, which demand a few computer resources and which are very effective for quadratic and approximately quadratic optimization under large and infinite n because of indifferences to dimension of a quadratic objective. These algorithms are based on new form of a necessary condition for optimality (NCO).

2 New interpretation of NCO

For smooth function $J(u)$ over R^n the classical local NCO, in an adjoint space, is of the following form:

$$\|\nabla J(u_*)\|_{R^*} = 0. \tag{4}$$

Let us remind that a space of gradients is an adjoint space in relation to a space of controls, i.e. $\nabla J \in R^*$.

In complicated problems, an exact value of optimal control is unattainable because of computing errors. In this case the condition (4) is useless, it is never implemented. The approximate implementation of (4) does not reveal proximity of control to the optimal value. NCO in the following form can be more useful [12]:

$$\nabla_i J(u^k) \to 0 \text{ under } u^k \to u_* \quad \forall i \in \{1, \ldots, n\} \text{ uniformly in } R^*,$$

where $\nabla_i J$ is i component of a vector ∇J. In the special case for the best (optimal) uniform convergence in adjoint space R^* we can formulate NCO in the following way.

Theorem 2.1 (NCO) *Let $J(u)$ be a smooth function, $u \in R^n$ and let u_* be a local minimizer. Then in some vicinity of u_* a sequence $u^k \to u_*$ exists with conditions*

$$\boxed{\begin{array}{ll} \dfrac{\nabla_i J^k}{\nabla_i J^{k-1}} = c^k, & i = 1, \dots, n, \quad k = 1, 2, ... \\ \\ \nabla_i J^0 \neq 0 \quad \forall i \in \{1, \dots, n\}, & J^k < J^{k-1}. \end{array}} \tag{5}$$

The number c^k is a constant for all i at iteration k. If $c^k = 0$, then $\nabla J^k = 0$ and we receive traditional NCO (4). If $c^k > 0$, then $\nabla J^k = c^k \nabla J^{k-1}$, the gradients ∇J^k and ∇J^{k-1} are codirectional at points u^k and u^{k-1}, here product $P = \left(\nabla J^k, \nabla J^{k-1}\right)_{R^*} > 0$. If $c^k < 0$, then $\nabla J^k = -\left|c^k\right| \nabla J^{k-1}$, the gradients are opposite and product $P < 0$.

It is obvious that for convex quadratic objective $J(u)$ the descent direction from u^k to u^{k-1} under conditions (5) will cross the optimal point u_*. Such descent direction coincides with the Newton direction. For an arbitrary objective $J(u)$ the descent trajectory satisfying (5) will not be a straight line to u_* because NCO (5) describes approximations to optimal control indirectly: we try to build a descent in control space R^n according to the information received in adjoint space R^*.

The classical NCO is a condition at stationary point of objective $J(u)$. The new NCO is a condition for descent trajectory. The peculiarities of NCO (5) are the following ones:

- it is a local condition for components of vector ∇J but not an integrated (normed) condition as the classical NCO;
- it is formulated not at point u_* but in its neighborhood.

These peculiarities are very important for iterative algorithms of descent to optimal point u_*.

3 Optimization algorithms with regulated direction of descent

For implementation of a descent to u_* according to NCO (5) we shall use the algorithm:

$$u^{k+1} = u^k - b^k \alpha^k \otimes \nabla J(u^k), \quad k = 0, 1, ..., \tag{6}$$

in a detailed form:

$$u_i^{k+1} = u_i^k - b^k \alpha_i^k \nabla_i J(u^k), \quad i = 1, ..., n, \quad k = 0, 1, ...,$$

where $\alpha^k \in R^*$, i.e. the parameter α^k is a vector in adjoint space R^*.

Algorithm (6) can realize NCO (5) if the parameter α^k 'correctly' regulates a minimization direction about a gradient ∇J^k. For this reason the author names the algorithms of a type (6) as *methods with regulated direction of descent*. Such algorithms do not require computing the matrixes as the Newton's type methods do.

It is necessary to say that the algorithms of type (6) — including traditional gradient methods under $\alpha^k \equiv 1$ — are regularization methods [13], they are stable to computing errors.

Now we discuss the two methods of choice of the parameter α^k for minimization of a quadratic J.

3.1 Optimal choice of parameter α^k

Given a convex quadratic objective $J(u)$ and initial guess u^0 such that $\nabla_i J^0 \neq 0 \ \forall i \in \{1, ..., n\}$. Let us find descent direction $d(u^0) = -\alpha \otimes \nabla J(u^0)$, which will cross the optimal control u_*. For that in the neighborhood of point u^0 we have to find an auxiliary point u^k satisfying NCO (5). This idea is shown on Fig.1 for $u \in R^2$.

Let point u^k be on δ-vicinity of u^0:

$$u^k = u^0 - \alpha^k \otimes \nabla J^0, \tag{7}$$

where α^k supplies a constant radius $\delta = \left\|u^k - u^0\right\| > 0$ and $J^k < J^0$. In (5) the constant c^k can be calculated by

$$c^k = \frac{\left\|\nabla J^k\right\|}{\left\|\nabla J^0\right\|} \mathrm{sign} P, \qquad P = \left(\nabla J^k, \nabla J^0\right)$$

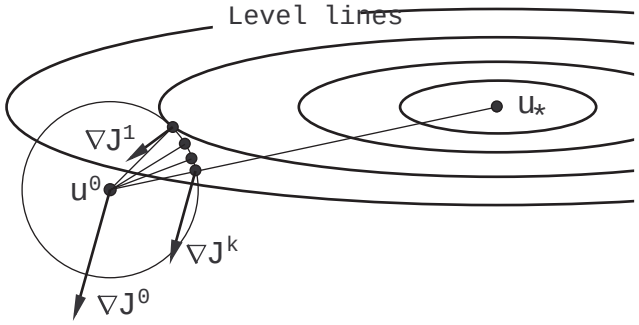

Figure 1: Minimization of a quadratic $J(u)$, $u \in R^2$ under optimal descent direction by NCO (8).

Then NCO (5) can be written in a form of nonlinear system, which we call *NCO-function* $\eta \in R^*$:

$$\eta_i(\alpha^k) \equiv \frac{\nabla_i J^k}{\nabla_i J^0} \frac{\left\|\nabla J^0\right\|}{\left\|\nabla J^k\right\|} - \mathrm{sign} P = 0, \quad i = 1, ..., n, \tag{8}$$

where gradient $\nabla J^k = \nabla J(\alpha^k)$, it depends from a vector α^k according to (7).

Finally we need to find the optimal parameter-vector α^k such that $\eta_i(\alpha^k) = 0 \ \forall i$. Parameter α^k, satisfying (8), gives us descent direction $d^0 = -\alpha^k \otimes \nabla J^0$ to point u_*. After satisfaction of (8) we need to do the unique iteration by method (6) under step-size (3):

$$u_* = u^0 - b^0 \alpha^k \otimes \nabla J^0. \tag{9}$$

Because of quadratic objective the dependence $J(b) = J(u^0 - b\alpha^k \otimes \nabla J^0)$ is a parabola, and the reader can verify that for any u^0 and $u^k \neq u^0$ the step-size is given by

$$b^0 = \left\|\nabla J^0\right\| / \left(\left\|\nabla J^0\right\| - \left\|\nabla J^k\right\| \mathrm{sign} P\right) . \tag{10}$$

The described method is similar to discrete Newton method in the following sense. Both methods compute the optimal direction of descent using a gradient in the neighborhood of point u^0. The methods differ from each other in the following way. Newton method requires beforehand known number of computation of gradients ∇J, namely $n \times n + 1$, it requires $n \times n$ cells of storage for received Hesse matrix, and requires the inverse of one that is usually replaced by calculation of a minimization step $s^0 \equiv b^0 d^0$ by solving a linear system $Hs^0 = -\nabla J^0$. The Hesse matrix gives not only optimal direction but also optimal depth of descent for a quadratic J.

The new method requires beforehand unknown number of computation of ∇J to find optimal minimizing direction $d^0 = -\alpha^k \otimes \nabla J^0$. The amount of computation depends on accuracy of solving the nonlinear system (8). This algorithm does not approximate the Hesse matrix. After solving NCO (8) by some inner iterations the method (9) requires one outer iteration under step-size (10).

The implementation of the NCO (8) essentially depends on method efficiency of solving the nonlinear system (8). For example, to vanish of NCO vector-function $\eta(\alpha)$, we are going to consider a simple iteration method:

$$\begin{aligned} \alpha_i^k &= \alpha_i^{k-1} + \tau^{k-1} \frac{\delta / \sqrt{n}}{|\nabla_i J^0|} \eta_i(\alpha^{k-1}), \quad k = 1, 2, ... \\ \alpha_i^0 &= \frac{\delta / \sqrt{n}}{|\nabla_i J^0|}, \quad i = 1, ..., n, \end{aligned} \tag{11}$$

where τ is a method step. Iterations (11) are inner ones relative to outer iteration (9). Do not forget to keep a δ -radius, i.e., to norm a vector α^k:

$$\alpha_i^k := \alpha_i^k \frac{\delta}{\|\alpha^k \otimes \nabla J^0\|} \tag{12}$$

The convergence of method (11) depends on a step τ. Initial value of τ^0 we shall select by the following requirement to angle γ between vectors $\alpha^0 \otimes \nabla J^0$ and $\alpha^1 \otimes \nabla J^0$: $5^\circ \le |\gamma| \le 60^\circ$. This requirement is convenient to check by a value of $\cos\gamma = \left(\alpha^0 \otimes \nabla J^0, \alpha^1 \otimes \nabla J^0\right) / \delta^2 = \sum_{i=1}^n \alpha_i^0 \alpha_i^1 \left(\nabla J^0\right)^2 \Big/ \delta^2$. It is necessary

$$0.5 \le \cos\gamma \le 0.996. \tag{13}$$

If the left-hand inequality is upset, then we have to decrease τ^0. If the right-hand inequality is upset, then we have to increase τ^0. Further we shall adjust a step τ by the condition:

$$\tau^{k-1} = \begin{cases} 0.3\tau^{k-2} & \text{if } \left(\eta^{k-1}, \eta^{k-2}\right)_{R^*} \le 0, \\ b_1\tau^{k-2} & \text{otherwise, } b_1 \ge 1 \end{cases} \qquad k = 2, 3, ... \tag{14}$$

For a stop of algorithm (11), let us introduce a precision N of computing α^k, where N is a number of digits of α^k in floating-point notation. For this reason the precision criterion for method (11) can be written as:

$$\text{if} \quad \alpha_i^k = \alpha_i^{k-1} \quad \forall i \in \{1, ..., n\} \quad \text{under precision } N, \text{ then there is a stop.}$$

Now we are going to write the whole algorithm for minimization of a quadratic objective function $J(u)$, $u \in R^n$.

Parameters. $\delta > 0$, $\beta_1 \ge 1$, N.
Data. Set and remember u^0.
Beginning of inner iterations, $k := 0$.
Step 1. Compute J, ∇J, $\|\nabla J\|$, if $\|\nabla J\| = 0$ then stop.
Step 2. If $k = 0$ then:
- if $\nabla_i J = 0$ for some i, then go to Data;
- remember J^0,∇J^0, $\left\|\nabla J^0\right\|$;
- compute initial $\alpha_i = \delta / \sqrt{n} / \left|\nabla_i J^0\right|$, set $\tau = 1$ and go to Step 7.

Step 3. If $k = 1$ and $J \ge J^0$, then $\delta := 0.3\delta$ and go to the Beginning of inner iterations.
Step 4. Compute $P = (\nabla J^0, \nabla J)$.
Step 5. Compute NCO function $\eta_i = \nabla_i J \left\|\nabla J^0\right\| / \nabla_i J^0 / \|\nabla J\| - \text{sign} P$.
Step 6. The simple iteration method.
If $k \ge 2$, then
- compute product $\left(\eta, \eta^{old}\right)$;
- If $\left(\eta, \eta^{old}\right) \le 0$, then $\tau := 0.1\tau$, else $\tau := \beta_1\tau$.

6a: Compute $\alpha_i := \alpha_i + \tau\eta_i\delta / \sqrt{n} / \left|\nabla_i J^0\right|$.
Norm vector α : $\alpha_i := \alpha_i \delta \Big/ \left\|\alpha \otimes \nabla J^0\right\|$.
If $\alpha_i = \alpha_i^{old}$ $\forall i$ under precision N, then go to Step 10.
If $k = 1$, then select initial τ:
- compute $\cos\gamma$;
- If $\cos\gamma < 0.5$, then $\tau := 0.3\tau$ and go to label **6a**;
- If $\cos\gamma > 0.996$, then $\tau := 2\tau$ and go to label **6a**.

Step 7. Remember: $\alpha^{old} := \alpha$, $\eta^{old} := \eta$.
Step 8. $k := k + 1$.
Step 9. Inner iteration: $u_i = u_i^0 - \alpha_i \nabla_i J^0$. Go to Step 1.
Step 10. Compute step-size $b = \left\|\nabla J^0\right\| / \left(\left\|\nabla J^0\right\| - \|\nabla J\| \text{sign} P\right)$.
Step 11. Outer iteration: $u_{*i} = u_i^0 - b\alpha_i \nabla_i J^0$.
Stop.

The described algorithm requires one computation of a function $J(u)$ and a gradient ∇J at each inner iteration of k. It uses only 1-dimensional arrays: u^0, ∇J^0, u, ∇J, η, η^{old}, α, α^{old}, i.e., for large n the algorithm demands approximately $8n$ memory cells. If you want to use a method more effective than the described simple iteration method, then replace Step 6 inside the algorithm.

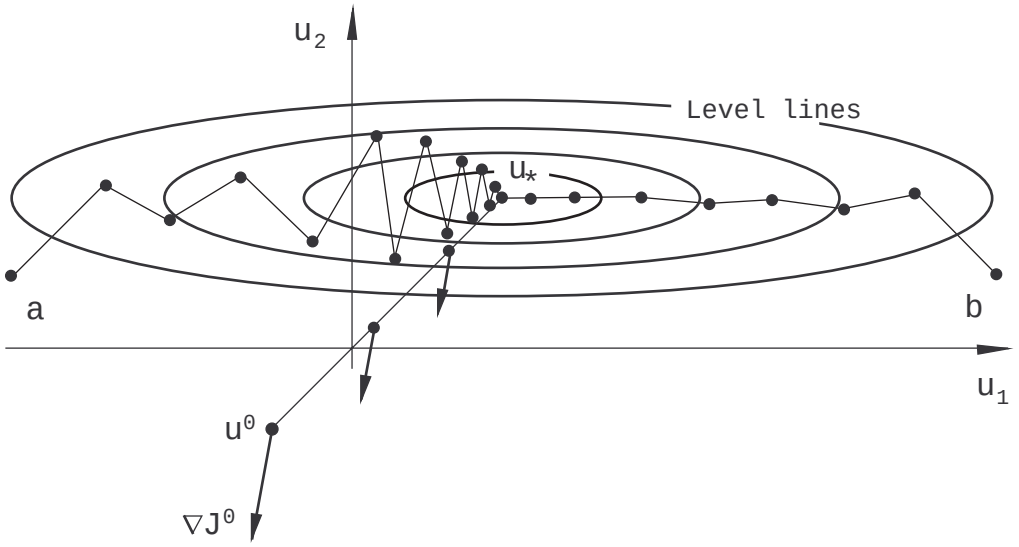

Figure 2: Minimization of a quadratic $J(u)$, $u \in R^2$ under heuristic descent direction by method (6). The point 'a': unsuccessful initial guess under α^k (15); The point u^0: successful initial guess under α^k (15); The point 'b': unsuccessful initial guess under α^k, which removes the oscillations according to (17).

3.2 Heuristic choice of parameter α^k

Given a convex quadratic objective $J(u)$ and initial guess u^0 such that $\nabla_i J^0 \neq 0 \ \forall i \in \{1, ..., n\}$. For algorithm (6) — $u^{k+1} = u^k - b^k \alpha^k \otimes \nabla J(u^k)$ — we set α^k under condition [12]:

$$\alpha_i^k = \alpha_i^0 = \frac{\delta / \sqrt{n}}{|\nabla_i J^0|}, \quad i = 1, \ldots, n \tag{15}$$

and set a step-size b^k under conditions: $b^0 = 1$, and

$$\begin{cases} \text{if } J^1 \geq J^0, & \text{then } \delta := 0.8\delta, \text{and repeat first iteration;} \\ \text{if } J^k < J^{k-1}, & \text{then } b^k = b_1 b^{k-1}, b_1 \geq 1, \ k \geq 1; \\ \text{if } J^k \geq J^{k-1}, & \text{then repeat previous iteration until } J^k < J^{k-1} \\ & \text{for } b^{k-1} = b_2 b^{k-2}, \ b_2 \in (0,1), \ k \geq 2. \end{cases} \tag{16}$$

The first step by method (6) with (15) will change the components u_i^0 by δ, i.e., it will give a first descent at 45° to all coordinate axes in space R^n. The heuristic parameter (15) equalizes a sensitivity of the minimization direction $d^0 = -\alpha^0 \otimes \nabla J^0$ to all components of the control. Besides, it prevents a descent from systematic computing errors of ∇J. At the first iteration this prevention is 100%.

Obviously, the parameter (15) will not satisfy NCO (5) for any initial guess u^0. If segment $[u^0, u_*]$ has angle 45° to all coordinate axes, then method (6) with α^k from (15) will satisfy NCO very well, and for steps under condition (16) the method will reach a minimum (see Fig.2). If segment $[u^0, u_*]$ has the angle about zero to some coordinate axes, then method (6) with α^k from (15) can lead to numerous ineffective oscillations. Look at the corresponding example on Fig.2, the initial point 'a'.

For removing the possible oscillations, due to unsuccessful initial guess, we can do the following correction of α^k (look at Fig.2, the initial point 'b'):

$$\alpha_i^k = \begin{cases} \alpha_i^{k-1} & \text{if } \operatorname{sign}\nabla_i J^k = \operatorname{sign}\nabla_i J^{k-1}, \\ b_3 \alpha_i^{k-1} & \text{otherwise, } b_3 \in (0,1), \end{cases} \qquad i = 1, ..., n, \quad k = 1, 2, ...$$

$$\alpha_i^0 = \frac{\delta / \sqrt{n}}{|\nabla_i J^0|}, \quad \nabla_i J^0 \neq 0 \quad \forall i \in \{1, ..., n\}, \tag{17}$$

We note that for any initial guess method (6) with (16),(17) is a minimization method, because the correction direction $d = -\alpha \otimes \nabla J$ (we recall that $\alpha_i^k > 0\ \forall i$) and the steepest descent direction $-\nabla J$ form an angle less than $90°$. For the first step we shall check equality $\nabla_i J^0 = 0$ approximately, i.e., we shall require $\left|\nabla_i J^0\right| \geq \varepsilon \quad \forall i$, where ε is a small number. A too small component $\nabla_i J^0$ will result in too large component α_i^0 that strongly reduces a step-size b and decelerates convergence.

Now we write wholly the minimization algorithm (6),(16),(17).

Parameters. $\delta > 0$, $\varepsilon \gtrsim 0$, $b_1 \geq 1$, $b_2 \in (0,1)$, $b_3 \in (0,1)$.
Data. Set initial u.
Beginning of iterations, $k := 0$.
Step 1. Compute J, ∇J, $\|\nabla J\|$, if $\|\nabla J\| = 0$, then stop.
Step 2. If $k = 0$ then:
- if $|\nabla_i J| < \varepsilon$ for some i, then go to Data;
- compute $\alpha_i = \delta/\sqrt{n}/\left|\nabla_i J^0\right|$, $b := 1$, and go to Step 7.

Step 3. If $J \geq J^{old}$ then:
- If $k = 1$, then $\delta := 0.5\delta$, $k := 0$, $u := u^{old}$, go to Step 1;
- If $k > 1$, then:
  - repeat:
    $b := b_2 b$, if $b = 0$, then stop;
    reiterate step: $u_i := u_i^{old} - b\alpha_i \nabla_i J^{old}$; compute J;
    stop repeat if $J < J^{old}$;
  - compute ∇J, and go to Step 7.

Step 4. Smooth over oscillations: If $\mathrm{sign}\nabla_i J \neq \mathrm{sign}\nabla_i J^{old}$, then $\alpha_i := b_3 \alpha_i$.
Step 5. Increase convergence: $b := b_1 b$.
Step 6. Remember: $u^{old} := u$, $\nabla J^{old} := \nabla J$, $J^{old} := J$.
Step 7. Compute next approximation: $u_i := u_i - b\alpha_i \nabla_i J$.
Step 8. $k := k + 1$, and go to Step 1.

To implement this algorithm we need the following 1-dimensional arrays: u, u^{old}, ∇J, ∇J^{old}, α, i.e., for large n the algorithm demands approximately $5n$ memory cells only. The advantage of method (6),(16),(17) on a comparison with SDM is essential descent to u_* for all components u_i. Traditional SDM has essential descent only on components with large $\nabla_i J$. Besides algorithm (16) demands rather a small number of calculations J as compared with (3). It is to be remembered that the proposed algorithm is weakly sensible to computing errors.

3.3 Heuristic algorithm based on Fletcher–Reeves' direction

Let us upgrade algorithm (6). We shall regulate a descent direction not concerning the gradient but concerning the Fletcher–Reeves' direction [5]:

$$u^{k+1} = u^k + b^k \alpha^k \otimes p(u^k), \quad k = 0, 1, \dots , \tag{18}$$

where $p^k = -\nabla J^k + \dfrac{\left\|\nabla J^k\right\|^2}{\left\|\nabla J^{k-1}\right\|^2} p^{k-1}$, $p^0 = -\nabla J^0$. We shall set the vector α^k as in section 3.2 from a condition of equalization of sensitivity of a descent direction for the first step. Then, we receive:

$$\alpha_i^k = \begin{cases} \alpha_i^{k-1} & \text{if sign } p_i^k = \text{sign } p_i^{k-1}, \\ b_3 \alpha_i^{k-1} & \text{otherwise, } b_3 \in (0,1), \end{cases} \qquad i = 1, \dots, n, \quad k > 0, \tag{19}$$

$$\alpha_i^0 = \frac{\delta/\sqrt{n}}{|\nabla_i J^0|}, \quad \nabla_i J^0 \neq 0 \quad \forall i \in \{1, \dots, n\}. \tag{20}$$

In expression (16) we shell change the last condition:

$$\begin{aligned} &\text{if } J^k \geq J^{k-1}, \quad \text{then repeat previous iteration until } J^k < J^{k-1} \\ &\qquad \text{for } p^{k-1} = -\nabla J^{k-1},\ b^{k-1} = b_2 b^{k-2},\ b_2 \in (0,1),\ k \geq 2. \end{aligned} \tag{21}$$

This condition removes accumulated computing errors in a vector p^k and guaranties minimization properties for algorithm (18).

Table 1: Iterations k and function/gradient evaluations K under minimization of J_1.

Dimension, n	2	5	10	50	100	300
SDM, k/K	4/144	$710/2.1\cdot10^3$	-	-	-	-
CGM, k/K	2/134	$23/1.3\cdot10^3$	$8.6\cdot10^3/4.7\cdot10^5$	-	-	-
DNM, k/K	1/6	1/27	1/102	$1/2.5\cdot10^3$	$1/10^4$	$1/9\cdot10^4$
RDDM-opt, k/K	1/7	1/11	1/13	1/21	1/15	1/29
RDDM-h, k/K	8/15	10/19	11/21	13/25	13/25	13/25
RDDM-FR, k/K	9/17	11/21	12/23	14/27	14/27	13/25

Table 2: Iterations k and function/gradient evaluations K under minimization of J_2.

Dimension, n	2	5	10	50	100	300
SDM, k/K	2/80	$174/5.2\cdot10^3$	$596/1.8\cdot10^4$	$8.2\cdot10^3/2.5\cdot10^5$	$6.8\cdot10^4/2\cdot10^6$	$5.6\cdot10^4/1.7\cdot10^6$
CGM, k/K	2/130	5/307	12/702	$101/5.6\cdot10^4$	$207/1.1\cdot10^5$	$478/2.6\cdot10^5$
DNM, k/K	1/6	1/27	1/102	$1/2.5\cdot10^3$	$1/10^4$	$1/9\cdot10^4$
RDDM-opt, k/K	1/53	1/47	1/45	1/59	1/59	1/61
RDDM-h, k/K	26/55	19/37	18/35	13/25	13/25	13/25
RDDM-FR, k/K	16/35	37/83	18/37	15/31	15/31	13/25

4 Testing

4.1 Finite-dimensional control

To test of the new algorithms we are going to consider three strictly convex quadratic functions with badly conditioned matrixes:

$$J_1 = \sum_{i=1}^{n} 10^{i-1}(u_i-1)^2;$$

$$J_2 = (u_2-1)^2 + \sum_{i=2}^{n} i(u_{i-1}+u_i-2)^2;$$

$$J_3 = (u_1+u_n-2)^2 + \sum_{i=2}^{n}(u_{i-1}-iu_i+i-1)^2;$$

These functions have a unique minimum at point $u_* = 1 \in R^n$.

You can download the test software at URL

ftp://ftp.dongu.donetsk.ua/pub/faculty/physical/kkt/tolstykh/min-test.arj

or via author Web page.

This software was prepared for MS DOS and has interactive graphic interface. You can input dimension n, function J, minimization method (including traditional methods), initial point u^0 and you will watch a descent track $\{u^k\}$ at function's level lines with normed gradients ∇J^k in section $\{u_1, u_2\}$. Moreover, function J^k, norm $\left\|\nabla J^k\right\|$ and deviation $\triangle u^k = \left\|u^k - u_*\right\|$ are displayed. All iterations are stopped for accuracy less than 1% of u_*. For method with optimal descent direction (optimal α) you can choose the precision N.

To estimate the minimization method efficiency, let us introduce a notion of amount of evaluations. In real optimal control problems the largest evaluations are needed for J^k and for ∇J^k. Usually computing J^k is approximately equal to computing ∇J^k. So evaluating J^k or ∇J^k we mean as one evaluation K.

Minimization results for the Steepest Descent Method (SDM), the Polak–Ribière Conjugate Gradient Method (CGM), the Discrete (finite difference derivative) Newton Method (DNM), the Regulated Direction of Descent Methods with optimal α^k (RDDM-opt), with heuristic α^k (RDDM-h) and heuristic method based on Fletcher–Reeves' direction (RDDM-FR) are shown for three functions in Tables 1–3. The initial point there was taken $u^0 = \{-0.8, -1.2, -1.4, ...\}$, where $u_i^0 = u_{i-1}^0 + (u_{i-1}^0 - u_{i-2}^0)/2,\ i = 3, ..., n$. For RDDM-opt k denotes outer iterations here, radius $\delta = 0.3$, coefficient $\beta_1 = 1.1$, and the precision N was changing from 2 to 4. For RDDM-h and RDDM-FR the data were: $\delta = 0.3$, $\varepsilon = 10^{-10}$, $b_1 = 1.3,\ b_2 = 0.8,\ b_3 = 0.3$.

Table 3: Iterations k and function/gradient evaluations K under minimization of J_3.

Dimension, n	2	5	10	50	100	300
SDM, k/K	7/234	$94/3 \cdot 10^3$	$3366/1 \cdot 10^4$	$5.6 \cdot 10^3/1.7 \cdot 10^5$	$1.4 \cdot 10^4/4.1 \cdot 10^5$	$1.2 \cdot 10^5/3.5 \cdot 10^6$
CGM, k/K	2/130	5/307	13/702	$75/5.6 \cdot 10^4$	$155/1.1 \cdot 10^5$	$470/2.6 \cdot 10^5$
DNM, k/K	1/6	1/27	1/102	$1/2.5 \cdot 10^3$	$1/10^4$	$1/9 \cdot 10^4$
RDDM-opt, k/K	1/29	1/157	1/275	1/321	1/301	1/331
RDDM-h, k/K	8/15	31/61	11/21	22/49	8/15	17/33
RDDM-FR, k/K	7/13	15/31	20/41	19/39	20/41	18/35

We see the new methods are indifferent to dimension of quadratic objective. The tested methods for large n use approximately following number of memory cells: SDM — $4n$, CGM — $6n$, QNM — $n \times n + 4n$, RDDM-opt — $8n$, RDDM-h — $5n$, RDDM-FR — $6n$.

4.2 Infinite-dimensional control

The new methods were generalized for control as a function. It is very easy because the described algorithms are not sensible to dimension of control. For example, for unsteady control in all conditions and methods we need to write $u(t)$ in please of u_i, $\nabla J(t;u)$ in please of $\nabla_i J(u)$, $t \in S$ in please of $i \in \{1,\dots,n\}$, where S is a time interval. The norm in R^n we will replace by norm in $L_2(S)$.

Under computer implementation of the infinite-dimensional optimization method we need to do a discretization of S, and instead t we will receive the points i of a finite-difference network. We will finally obtain the algorithms, which coincide formally with the algorithms described in section 3. The distinction is contained in impossibility to smooth over the infinite-dimensional oscillations. From that we must not use formulas (17),(19).

The new methods were applied to optimal control and identification problems for parabolic and hyperbolic equations [1],[2],[14],[16]. You can download a test software for Windows operating system at URL

ftp://ftp.dongu.donetsk.ua/pub/faculty/physical/kkt/tolstykh/setup.exe

or via author Web page.

The first problem is a simple control problem about a thermal flow in the chemical reactor through a steel wall. The corresponding parabolic system on $[t_a, t_b] \times [x_0, x_1]$ has the form:

$$C\rho\frac{\partial T}{\partial t} - \kappa\frac{\partial^2 T}{\partial x^2} = 0, \qquad \kappa\left.\frac{\partial T}{\partial x}\right|_{x_0} = q, \quad \kappa\left.\frac{\partial T}{\partial x}\right|_{x_1} = u(t), \quad T(t_a, x) = T_a, \tag{22}$$

where $T(t,x)$ is a temperature in the wall, C, ρ, κ is a heat capacity, density and heat conducting coefficients accordingly, q is a heat flow given by the chemical reaction.

The chemical reaction must be under temperature $T_*(t, x_0)$. So we have to find a heat flow $u(t)$, which will minimize the objective function

$$J(u) = \int_{t_a}^{t_b} (T - T_*)^2\big|_{x_0}\, dt. \tag{23}$$

The gradient of the objective function (23) is

$$\nabla J(t;u) = -f(t, x_1),$$

where f satisfies to adjoint problem on $[t_a, t_b] \times [x_0, x_1]$:

$$C\rho\frac{\partial f}{\partial t} + \kappa\frac{\partial^2 f}{\partial x^2} = 0, \qquad \kappa\left.\frac{\partial f}{\partial x}\right|_{x_0} = 2\,(T - T_*)\big|_{x_0}, \quad \kappa\left.\frac{\partial f}{\partial x}\right|_{x_1} = 0, \quad f(t_b, x) = 0. \tag{24}$$

The test is organized by the following way. You input a control $u_*(t)$ by sliders, the program computes the corresponding temperature T_* and a chosen optimization method starts from initial guess $u^0 = 400\ kJ/m^2 s$.

On Fig.3 are shown the controls received by CGM and RDDM-FR for 100 and 1000 iterations. The traditional method does not achieve u_* even for 1000 iterations.

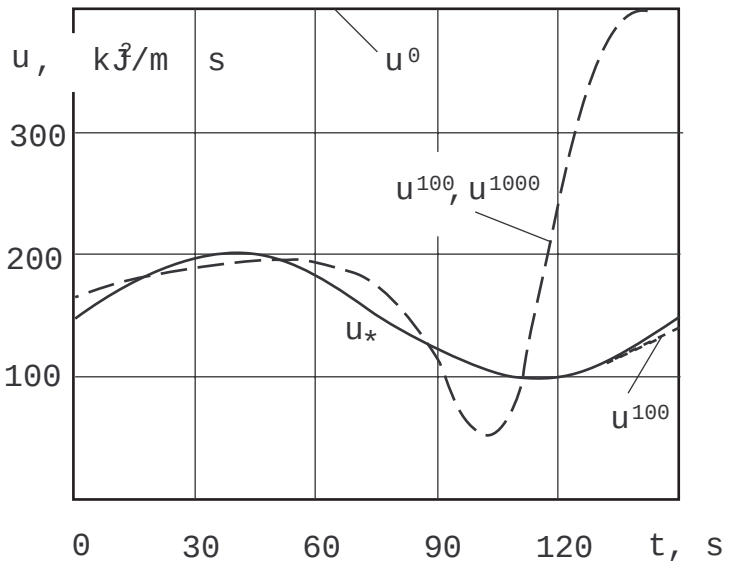

Figure 3: Minimization of function (23). Small dashed — regulated direction of descent method (18),(20),(21). Large dashed — conjugate gradient method.

5 Conclusions

From Tables 1–3 and Fig.3 we see that for large-dimensional quadratic optimization the regulated direction of descent methods are most effective. They use only first derivative of the objective function, they demonstrate indifferences to dimension of the problem, and they ensure the high accuracy of optimization under simple algorithms and small computer resources. The heuristic methods can be applied to approximately quadratic objective functions. For infinite-dimensional case the new methods have indisputable advantage.

References

[1] G.A. Atanov and S.T. Voronin, V.K. Tolstykh, *On parameter identification problem of open channel (Russian)*, Water Resources, Moscow, 4 (1986), pp. 69-78.

[2] V.S. Borodin and N.A. Volodin, V.K. Tolstykh, *Identification of parameters in forming of the casting models (Russian)*, Casting process, Kiev, 1 (1995), pp. 96-101.

[3] W. C. Davidon, *Variable Metric Method for Minimization*, J. Optim. 1 (1991), 1–17.

[4] J. E. Dennis, JR and Robert B. Schnabel, *Numerical methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Prentice–Hall, Inc., Englewood Cliffs, NJ, 1983.

[5] R. Fletcher and C. M. Reeves, Function Minimization by Conjugate Gradients, Comput J., 7 (1964), 149–154.

[6] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., John Wiley & Sons, New York, 1987.

[7] C. T. Kelley, *Iterative Methods for Optimization*, SIAM, Philadelphia, 1999.

[8] E. Polak, *Optimization: Algorithms and Consistent Approximations*, Springer–Verlag, New York, 1997.

[9] J. Nocedal and S. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.

[10] E. Polak and G. Ribière, *Note Sur la Convergence de Methodes de Directions Conjugèes*, Rev Francaise Informat Recherche Operationelle, 3e annèe, 16 (1969), 35–43.

[11] M. J. D. Powell. *Some Global Convergence Properties of a Variable Metric Algorithm Without Exact Line Searches*, in R. Cottle and C. Lemke, Nonlinear Programming, AMS, Providence, RI , 1976, 53–72.

[12] V. K. Tolstykh, *Direct Extreme Approach for Optimization of Distributed Parameter Systems (Russian)*, Yugo–Vostok, Donetsk, 1997.

[13] V. K. Tolstykh, *Applying the gradient method to optimization problems of the distributed parameter systems (Russian).* J. of Calcul. Mathem. and Mathem. Physics, vol.26, 1 (1986), pp. 137-140.

[14] V. K. Tolstykh and N.A Volodin, *Optimal control by heat flow in continuous casting of steel*, Operations Research Proc., Braunschweig: Springer,1996, pp. 480-483.

[15] F. P. Vasiliev, *Computing methods for solving of extreme problems (Russian)*, Nauka, Moscow, 1980.

[16] Volodin N.A. and V.K. Tolstykh, *Applying of the gradient optimization method to control problem in heat reactor (Russian)*, Automatica, Kiev, 1 (1993), pp. 40-44.