A Numerical Construction Algorithm of Nash and Stackelberg Solutions for Two-person Non-zero Sum Linear Positional Differential Games [⋆]

Anatolii F. Kleimenov [*] **Sergei I. Osipov** [**] **Dmitry R. Kuvshinov** [***]

[*] *Inst. of Math. and Mech., Ural Branch of RAS, 16, S.Kovalevskaja street, Ekaterinburg, 620219, Russia (e-mail: kleimenov@imm.uran.ru).*
[**] *Ural State University, 51, Lenin Ave., Ekaterinburg, 620083, Russia (e-mail: sergei.osipov@usu.ru)*
[***] *Ural State University, 51, Lenin Ave., Ekaterinburg, 620083, Russia (e-mail: evetro@2-u.ru)*

Abstract: The report evolves a method, which uses the formalization and results of positional antagonistic differential games theory, developed by N. N. Krasovskii and his scientific school, for constructing solutions of a class of non-antagonistic differential games. The method transforms non-antagonistic game into so-called non-standard optimal control problem. Numerical solutions for Stackelberg games are constructed by an algorithm developed by S. Osipov. Numerical Nash solution construction algorithm based upon auxiliary bimatrix games sequence is presented. Used computational geometry algorithms include convex hull construction, union and intersection of polygons and a Minkowski sum for polygons. Results of numerical experiment for a material point motion in plane are presented. The point is moved by force formed by two players. Every player has his personal target point. Among the obtained results, there is a Nash solution such that along the corresponding trajectory the position of the game is non-antagonistic, at first, and then becomes globally antagonistic starting from some moment of time.

Keywords: Differential games, Stackelberg games, Nash games, Trajectories, Computing, Algorithms

1. INTRODUCTION

There are various approaches for computation of solutions in differential games, see, e.g., Basar and Olsder (1982); Krasovskii and Subbotin (1988); Krasovskii and Krasovskii (1995); Krasovskii (1985); Kleimenov (1993).Many of them suggest numeric methods for solution computation. Such algorithms proposed for antagonistic games are, for example, discussed in papers Isakova et al. (1984); Vahrushev et al. (1987), as well as in other studies of the same and other authors. Comparing to this, there are distinctly less studies concerning non-antagonistic games and they usually deal with linear quadratic games. The present paper describes an algorithm for Nash equilibrium solutions and Stackelberg solutions in linear differential game with geometrical constraints for players' controls and terminal cost functionals of players. Algorithm and the program for Nash solutions was mainly implemented by D. Kuvshinov.

The paper is organized as follows. Section 2 contains problem statement. Section 3 describes common method for Nash and Stackelberg solutions construction based on reduction of original problem to non-standard problems of (optimal) control. Section 4 presents description of two algorithms. The first one builds a Nash solution with the help of auxiliary bimatrix game. The second one solves Stackelberg problem approximating admissible trajectories via repetitive intersections of stable bridges and local attainability sets. Brief description of the program implementation is given in Section 5. Results of numerical experiment for a material point motion in plane are presented in Section 6. The point is moved by force formed by two players. Every player has his personal target point. Among the obtained results, there is a Nash solution such that along the corresponding trajectory the position of the game is non-antagonistic, at first, and then becomes globally antagonistic starting from some moment of time. Finally, Section 7 proposes possible study perspectives.

2. PROBLEM STATEMENT

Let the dynamics of a two-person non-antagonistic positional differential game be described by the equation

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{C}(t)\mathbf{v}(t),$$
$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad t \in [t_0, \theta], \tag{1}$$

[⋆] The report partially supported by Russian Foundation for Basic Research, grant 09–01–00313.

where $\mathbf{x} \in \mathbb{R}^n$ is a phase vector. Matrices $\mathbf{A}(t)$, $\mathbf{B}(t)$ and $\mathbf{C}(t)$ are continuous and have dimensions $n \times n$, $n \times k$ and $n \times l$, respectively. Controls $\mathbf{u} \in P \subset \mathbb{R}^k$ and $\mathbf{v} \in Q \subset \mathbb{R}^l$ are handled by Player 1 and Player 2, respectively. Sets P and Q are assumed to be convex polyhedrons. The final time θ is fixed.

Suppose that Player 1 goal is maximization of the cost functional $\sigma_1(\mathbf{x}(\theta))$, while Player 2 maximizes the cost functional $\sigma_2(\mathbf{x}(\theta))$, where functions $\sigma_1 : \mathbb{R}^n \to \mathbb{R}$ and $\sigma_2 : \mathbb{R}^n \to \mathbb{R}$ are continuous and concave.

It is also assumed that both the players know value $\mathbf{x}(t)$ at the current moment of time $t \in [t_0, \theta)$. Then formalization of players' strategies in the game could be based on the formalization and results of positional antagonistic differential games theory (Krasovskii (1985); Krasovskii and Subbotin (1988)). According to this formalization (see also Kleimenov (1993)), strategies are described as belonging to the class of pure positional strategies and are equated to pairs of functions. Strategy of Player 1 U is equated to a pair $\{\mathbf{u}(t, \mathbf{x}, \epsilon), \beta_1(\epsilon)\}$, where $\mathbf{u}(\cdot, \cdot, \cdot)$ is an arbitrary function of position $(t, \mathbf{x})$ and a positive precision parameter ϵ and having values in the set P. The function $\beta_1 : (0, \infty) \to (0, \infty)$ is a continuous monotone one and satisfies the condition $\beta_1(\epsilon) \to 0$ if $\epsilon \to 0$. The function $\beta_1(\cdot)$ has the following sense. For a fixed ϵ the value $\beta_1(\epsilon)$ is the upper bound for the step of a subdivision of the interval $[t_0, \theta]$, which Player 1 uses for forming step-by-step motions. Strategy $V \div \{\mathbf{v}(t, \mathbf{x}, \epsilon), \beta_2(\epsilon)\}$ of Player 2 is defined analogously. A motion $\mathbf{x}[t; t_0, \mathbf{x}_0, U, V]$, generated by a pair of strategies (U, V), is introduced. The set of motions $X(t_0, \mathbf{x}_0, U, V)$ is non-empty (Krasovskii (1985)).

2.1 Nash solutions

A pair of strategies (U^N, V^N) is called a *Nash equilibrium solution* (*N-solution*) of the game, if for any motion $\mathbf{x}^*[\cdot] \in X(t_0, \mathbf{x}_0, U^N, V^N)$, any $\tau \in [t_0, \theta]$ and any strategies U and V the following inequalities are held

$$\begin{aligned} &\max \sigma_1(\mathbf{x}[\theta; \tau, \mathbf{x}^*[\tau], U, V^N]) \leq \\ &\qquad \min \sigma_1(\mathbf{x}[\theta; \tau, \mathbf{x}^*[\tau], U^N, V^N]), \\ &\max \sigma_2(\mathbf{x}[\theta; \tau, \mathbf{x}^*[\tau], U^N, V]) \leq \\ &\qquad \min \sigma_2(\mathbf{x}[\theta; \tau, \mathbf{x}^*[\tau], U^N, V^N]). \end{aligned} \tag{2}$$

The operations of min and max in (2) are taken in the sets of corresponding motions.

Trajectories of the system (1) generated by N-solution are called N-trajectories.

2.2 Stackelberg solutions

Denote by $S1$ the game under the following assumptions. Player 1 chooses his strategy U before the game and informs Player 2 about this choice. Player 2 knowing the announced strategy U chooses a *rational strategy* V, providing maximization of cost functional σ_2. Player 1 is a leader here and Player 2 is a follower. Such a game is called a *hierarchical* one. The problem is to find a Player 1 strategy U^{S1}, which paired with the rational Player 2 strategy V^{S1} provides the maximum of the cost functional σ_1. Then a pair of strategies (U^{S1}, V^{S1}) is called Stackelberg solution ($S1$-solution) in hierarchical game with first player as a leader.

Denote by $S2$ a game, where Player 2 acts as a leader and Player 1 is a follower. This game is analogous to $S1$ game. We define Stackelberg solution ($S2$-solution) a pair (U^{S2}, V^{S2}) in hierarchical game with second player as a leader.

A trajectory, generated by Si-solution is called Si-trajectory. The task of the report is algorithm and program development for N- and Si-solutions construction.

3. NON-STANDARD PROBLEMS OF CONTROL

At first consider antagonistic positional differential games Γ_1 and Γ_2. The dynamics of both games is described by the equation (1). In the game Γ_i, Player i maximizes his cost functional $\sigma_i(\mathbf{x}(\theta))$, and Player $3 - i$ counter-acts to this goal. It is known from Krasovskii (1985); Krasovskii and Subbotin (1988), that both games Γ_1 and Γ_2 have universal saddle points

$$\{\mathbf{u}^{(i)}(t, \mathbf{x}, \epsilon), \mathbf{v}^{(i)}(t, \mathbf{x}, \epsilon)\}, \quad i = 1, 2 \tag{3}$$

and continuous value functions

$$\gamma_1(t, \mathbf{x}), \quad \gamma_2(t, \mathbf{x}). \tag{4}$$

It was shown in Kleimenov (1993), that the problem of finding N- and Si-solutions could be reduced to finding solution in following non-standard problems of control.

Problem 1. It needs to find a pair of measurable controls $\mathbf{u}(t)$ and $\mathbf{v}(t)$, $t_0 \leq t \leq \theta$, guaranteeing the fulfillment of conditions

$$\gamma_i(t_*, \mathbf{x}(t_*)) \leq \gamma_i(t^*, \mathbf{x}(t^*)), \quad i = 1, 2, \tag{5}$$

where $t_* \in [t_0, \theta)$ and $t^* \in (t_*, \theta]$.

Problem 2.i (i=1,2). It needs to find a pair of measurable functions $\mathbf{u}(t), \mathbf{v}(t)$, $t_0 \leq t \leq \theta$, providing the maximum of cost functional $\sigma_i(\mathbf{x}(\theta))$, holding following condition

$$\gamma_{3-i}(t, \mathbf{x}(t)) \leq \gamma_{3-i}(\theta, \mathbf{x}(\theta)) = \sigma_{3-i}(\mathbf{x}(\theta)) \tag{6}$$

for $t \in [t_0, \theta)$.

Here functions γ_1 and γ_2 are defined in (4).

Let piecewise continuous functions $\mathbf{u}^*(t)$ and $\mathbf{v}^*(t)$, $t_0 \leq t \leq \theta$ generate a trajectory $\mathbf{x}^*(t)$, $t_0 \leq t \leq \theta$ of the system (1). Consider strategies of Player 1 and Player 2 $U^{\text{o}} \div \{\mathbf{u}^{\text{o}}(t, \mathbf{x}, \epsilon), \beta_1^{\text{o}}(\epsilon)\}$ and $V^{\text{o}} \div \{\mathbf{v}^{\text{o}}(t, \mathbf{x}, \epsilon), \beta_2^{\text{o}}(\epsilon)\}$, where

$$\mathbf{u}^{\text{o}}(t, \mathbf{x}, \epsilon) = \begin{cases} \mathbf{u}^*(t) & \text{if } \|\mathbf{x} - \mathbf{x}^*(t)\| < \epsilon\varphi(t), \\ \mathbf{u}^{(2)}(t, \mathbf{x}, \epsilon) & \text{if } \|\mathbf{x} - \mathbf{x}^*(t)\| \geq \epsilon\varphi(t), \end{cases} \tag{7}$$

$$\mathbf{v}^{\text{o}}(t, \mathbf{x}, \epsilon) = \begin{cases} \mathbf{v}^*(t) & \text{if } \|\mathbf{x} - \mathbf{x}^*(t)\| < \epsilon\varphi(t), \\ \mathbf{v}^{(1)}(t, \mathbf{x}, \epsilon) & \text{if } \|\mathbf{x} - \mathbf{x}^*(t)\| \geq \epsilon\varphi(t), \end{cases} \tag{8}$$

for all $t \in [t_0, \theta]$. The functions $\beta_i(\cdot)$ and the positive increasing function $\varphi(\cdot)$ are chosen so that the inequality

$$||\mathbf{x}(t, t_0, x_0, U^{\text{o}}, \epsilon, \Delta_1, V^{\text{o}}, \epsilon, \Delta_2) - \mathbf{x}^*(t)|| < \epsilon\varphi(t) \tag{9}$$

holds for $\epsilon > 0$, $\delta(\Delta_i) \leq \beta_i(\epsilon)$, $t \in [t_0, \theta]$.

Usually in literature strategies $\mathbf{u}^{(2)}(t, \mathbf{x}, \epsilon), \mathbf{v}^{(1)}(t, \mathbf{x}, \epsilon)$ are called punishment strategies . The following theorem is true.

Theorem 1. Let controls $\mathbf{u}^*(\cdot)$ and $\mathbf{v}^*(\cdot)$ be a solution of Problem 1 (or Problem 2.i). Then the pair of strategies $(U^{\mathrm{o}}, V^{\mathrm{o}})$ (7)–(9) is an N-solution (or an *Si*-solution).

Generally it is very difficult to find the whole set of solutions for the non-standard problem described above. Therefore the report presents an algorithm for constructing only some N-solution. The algorithm is essentially constructing N-trajectories.

Remark 2. In Kleimenov (1997) a classification of positions $(t, \mathbf{x})$ in non-antagonistic positional differential game is proposed. Three types of positions: non-antagonistic, locally antagonistic, globally antagonistic were introduced. Dynamics of position type change along N-trajectories is of special interest.

4. ALGORITHMS

Suggested algorithms for finding solutions are based upon computational geometry algorithms. Unfortunately, at present time this procedures are only two-dimensional, so it is additionally assumed, that cost functionals of players σ_i depend on two selected components solely. Because of the fact all the problems considered could be reduced to problems in plane.

4.1 Algorithm for N-solution

The proposed algorithm is based on the procedure (see Kleimenov (1997)), which uses the principle of non-decrease of player payoffs, maximal shift in the direction best for one and another player and Nash equilibrium in auxiliary bimatrix games made up for each step in a subdivision of the time interval.

The procedure implies that Player i is interested in increasing the function $\gamma_i(t, \mathbf{x})$ along the trajectory, that is a solution, as in (5).

Suppose a position $(t, \mathbf{x})$ is given. We fix $\epsilon > 0$ and put $\tau(t, \epsilon) = \min\{t + \epsilon, \theta\}$. Denote by $w^1(\tau(t, \epsilon))$ and $w^2(\tau(t, \epsilon))$ the maximum points for functions $\gamma_1(t, \mathbf{x})$ and $\gamma_2(t, \mathbf{x})$, correspondingly, in the $\tau(t, \epsilon)$-neighborhood of some position $\mathbf{x}$.

Consider vectors

$$\mathbf{s}^1(\tau, \mathbf{x}, \epsilon) = w^1(\tau(t, \epsilon)) - \mathbf{x},$$
$$\mathbf{s}^2(\tau, \mathbf{x}, \epsilon) = w^2(\tau(t, \epsilon)) - \mathbf{x}.$$

We define vectors $\mathbf{u}_{10}(t, \mathbf{x}, \epsilon)$, $\mathbf{v}_{10}(t, \mathbf{x}, \epsilon)$, $\mathbf{u}_{20}(t, \mathbf{x}, \epsilon)$ and $\mathbf{v}_{20}(t, \mathbf{x}, \epsilon)$ from conditions

$$\max_{\mathbf{u}\in P, \mathbf{v}\in Q} \mathbf{s}^{1\top}[\mathbf{B}(t)\mathbf{u} + \mathbf{C}(t)\mathbf{v}] = \mathbf{s}^{1\top}[\mathbf{B}(t)\mathbf{u}_{10} + \mathbf{C}(t)\mathbf{v}_{10}],$$
$$\max_{\mathbf{u}\in P, \mathbf{v}\in Q} \mathbf{s}^{2\top}[\mathbf{B}(t)\mathbf{u} + \mathbf{C}(t)\mathbf{v}] = \mathbf{s}^{2\top}[\mathbf{B}(t)\mathbf{u}_{20} + \mathbf{C}(t)\mathbf{v}_{20}],$$
$$\gamma_1(t, \mathbf{x}(t)) \le \gamma_1(\tau(t, \epsilon), \mathbf{x}[\tau(t, \epsilon); t, \mathbf{x}(t), \mathbf{u}_{20}, \mathbf{v}_{20}]),$$
$$\gamma_2(t, \mathbf{x}(t)) \le \gamma_2(\tau(t, \epsilon), \mathbf{x}[\tau(t, \epsilon); t, \mathbf{x}(t), \mathbf{u}_{10}, \mathbf{v}_{10}]). \quad (10)$$

Two last inequalities come from the condition of non-decrease of functions $\gamma_i(\cdot, \cdot)$ along a motion, and maximums are searched only for vectors that hold these inequalities.

Now we construct an auxiliary bimatrix 2×2 game (A, B), in which the first player has two strategies: "to choose $\mathbf{u}_{10}$" and "to choose $\mathbf{u}_{20}$". Similarly, the second player has two strategies: "to choose $\mathbf{v}_{10}$" and "to choose $\mathbf{v}_{20}$". The payoff matrices of the players are defined as follows:

$$A = \left\| \begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{matrix} \right\|, \qquad B = \left\| \begin{matrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{matrix} \right\|,$$

$$a_{ij} = \gamma_1(\tau(t, \epsilon), \mathbf{x} + \delta(\mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u}_{i0} + \mathbf{C}(t)\mathbf{v}_{j0})),$$
$$b_{ij} = \gamma_2(\tau(t, \epsilon), \mathbf{x} + \delta(\mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u}_{i0} + \mathbf{C}(t)\mathbf{v}_{j0})),$$
$$i, j = 1, 2, \qquad \delta = \tau(t, \epsilon) - t.$$

Bimatrix game (A, B) has at least one Nash equilibrium in pure strategies. It is possible to take a Nash equilibrium as a controls for both players for an interval $(t, \tau(t, \epsilon)]$. Such an algorithm of players' controls construction generates a trajectory, which is an *N-trajectory.* When (A, B) has two equilibria (11 and 22), then one is chosen after the other was, and vice versa.

An algorithm, which takes a solution of (A, B) and tries to fit both players' controls to maximize shift along the direction of this solution, while holding inequalities in (10), is called *BM-procedure* here. To provide Nash equilibrium in the game, controls generated by BM-procedure (giving *BM-trajectory*) must be paired with punishment strategies. In this case, each player watches for trajectory and applies the punishment strategy, when other one evades following BM-trajectory. BM-procedure usually gives better results for both players than original (A, B) game based algorithm, but players have to agree to follow BM-trajectory before the game will start.

4.2 Algorithm for S1-solution

The general idea for the algorithm is to search $\max_\alpha \sigma_1(\mathbf{x}_\alpha[\theta])$, where $\mathbf{x}_\alpha[\theta]$ are node points of a grid constructed for a *set of admissible trajectories final states* D_1. This $\mathbf{x}_\alpha[\theta]$ serves an endpoint for S1-trajectory, which is then built back in time (controls $\mathbf{u}(t)$, $\mathbf{v}(t)$ may be found simultaneously). The D_1 set approximation is constructed by a procedure, that builds sequences of attainability sets (step-by-step in time), repeatedly throwing out the positions that do not satisfy (6). The procedure is described in brief below. More details (some designations differ) about it could be found in Kleimenov and Osipov (2003).

A special construction from the theory of antagonistic positional differential games called *stable bridge in pursuit-evasion game.* The aim of the follower in this game is to drive the phase vector to Lebesgue's set for the level function (for a chosen constant c) of his cost functional. (In order to find positions satisfying the inequality $\gamma_2(t, \mathbf{x}) \le c$.) Note, that any position at the bridge holds the inequality, but positions outside the bridge do not.

$\widetilde{W}^c_{1,t}$ designates an approximation of a bridge (in the pursuit-evasion game) section in the time moment $t =$ const. The following discrete scheme is used for building *admissible trajectories pipe* G^c_1 section approximation $\widetilde{G}^c_{1,t_k}$ (here k runs through some time interval subdivision, $k = 0$ corresponds to t_0 moment and $k = N$ corresponds to θ):

$$\widetilde{G}^c_{1,t_{k+1}} = \left[\widetilde{G}^c_{1,t_k} \oplus \delta_k(\mathbf{B}(t_k)P \oplus \mathbf{C}(t_k)Q)\right] \setminus \widetilde{W}^c_{1,t_{k+1}}, \quad (11)$$

where $\widetilde{G}^c_{1,t_0} = \{\mathbf{x}_0\}$, $\delta_k = t_{k+1} - t_k$. Operation $A \oplus B = \{a + b \mid a \in A, b \in B\}$ denotes Minkowski sum of two sets A and B.

Using this scheme, we iterate through c to make up D_1 of sequence of $D^c_1 = \widetilde{G}^c_{1,\theta} \cap \widetilde{W}^c_{1,\theta}$ in the following procedure:

```
(1) We have some initial step value $\delta c > 0$ constrained by $\delta c_{\min} \le \delta c$;
(2) let $D_1 = \varnothing$, $c = c^{\max} = \max_{\mathbf{x}\in\mathbb{R}^n} \sigma_2(\mathbf{x})$;
(3) build a pipe $\widetilde{G}^c_1$ and a set $D^c_1$ as in (11);
(4) supplement $D_1 := D_1 \cup \{(\mathbf{x}, c) \mid \mathbf{x} \in D^c_1\}$;
(5) if $\delta c \ge \delta c_{\min}$ then we choose next $c$ value:
      • if $\mathbf{x}_0 \in \widetilde{W}^c_{1,t_0}$ then a) return to the previous value $c := c + \delta c$, b) decrease step $\delta c$;
      • take next value $c := c - \delta c$;
      • go to item 2;
(6) quit.
```

D_2 set for $S2$ game may be built in the same way.

One example of S-trajectories numerical computation results was presented in Kleimenov et al. (2006). Program, used for value function calculation, is based on results Isakova et al. (1984); Vahrushev et al. (1987).

5. PROGRAM IMPLEMENTATION

A completely new approach to program implementation was successfully applied at first to N-solution computation and after that — to S-solutions. Computational geometry algorithms library, developed by S. Osipov in Fortran somewhere in 80's, became outdated, so it was temporarily substituted with a C++ wrapper library, which builds upon polygon tesselation facilities from *OpenGL Utility Library* (*GLU*). GLU functions and structures for polygonal primitives are intensively used by algorithms, mentioned herein. Examples of the next section were obtained with GLU being used for polygons processing.Advantages of GLU include straightforward API in C, which is simple to use in almost any programming environment. Many implementations exist (usually bundled with operational system), both proprietary and open source.

Despite positive results achieved by the implementation, another library (which is an open source project) was tested, as a possible future base for our algorithms. It was *Computational Geometry Algorithms Library* (*CGAL*), which goal "is to provide easy access to efficient and reliable geometric algorithms in the form of a C++ library" (see http://www.cgal.org/). While GLU is a convenient external component, CGAL provides a complex framework to expand upon and is not bounded by hardware-supported double precision arithmetics.

In the case of S-solutions, OpenMP was adopted (discrete scheme (11) is run for different c values in parallel). Tests on a machine with Intel Core 2 Duo processor demonstrated twofold run-time improvement for two-threaded computations against one-threaded.

6. AN EXAMPLE

The following vector equation

$$\ddot{\xi} = \mathbf{u} + \mathbf{v}, \quad \xi(t_0) = \xi_0, \quad \dot{\xi}(t_0) = \dot{\xi}_0 \quad (12)$$
$$\xi, \mathbf{u}, \mathbf{v} \in \mathbb{R}^2, \quad ||\mathbf{u}|| \le \mu, \quad ||\mathbf{v}|| \le \nu, \quad \mu > \nu$$

describes the motion of a material point of unit mass on the plane (ξ_1, ξ_2) under the action of a force $\mathbf{F} = \mathbf{u} + \mathbf{v}$. Player 1 (Player 2), who governs the control $\mathbf{u}$ ($\mathbf{v}$), tends to lead the material point as close as possible to the given target point $a^{(1)}$ ($a^{(2)}$) at the moment of time θ. Then players' cost functionals are

$$\sigma_i(\xi(\theta)) = -||\xi(\theta) - a^{(i)}||, \quad (13)$$
$$\xi = (\xi_1, \xi_2), \quad a^{(i)} = (a^{(i)}_1, a^{(i)}_2), \quad i = 1, 2,$$

where θ is final time.

By setting $y_1 = \xi_1$, $y_2 = \dot{\xi}_1$, $y_3 = \xi_2$, $y_4 = \dot{\xi}_2$ and making the following change of variables $x_1 = y_1 + (\theta - t)y_3$, $x_2 = y_2 + (\theta - t)y_4$, $x_3 = y_3$, $x_4 = y_4$ we get a system, which first and second equations are

$$\dot{x}_1 = (\theta - t)(u_1 + v_1), \quad (14)$$
$$\dot{x}_2 = (\theta - t)(u_2 + v_2).$$

Further, (13) can be written

$$\sigma_i(\mathbf{x}(\theta)) = -||\mathbf{x}(\theta) - a^{(i)}||, \quad \mathbf{x} = (x_1, x_2), \quad i = 1, 2. \quad (15)$$

Since the cost functional (15) depends on variables x_1 and x_2 only and the right-hand side of (14) does not depend on other variables, one can conclude, that it is sufficient to consider only reduced system (14) with cost functionals (15).

Then initial conditions for (14) are given by formulae

$$x_i(t_0) = x_{0i} = \xi_{0i} + (\theta - t_0)\dot{\xi}_{0i}, \quad i = 1, 2.$$

It can easily be shown, that value functions in antagonistic differential games Γ_1 and Γ_2 are given by formulae

$$\gamma_1(t, \mathbf{x}) = -||\mathbf{x} - a^{(1)}|| - \frac{(\theta - t)^2}{2}(\mu - \nu),$$
$$\gamma_2(t, \mathbf{x}) = \min\{-||\mathbf{x} - a^{(2)}|| + \frac{(\theta - t)^2}{2}(\mu - \nu), 0\}$$

and universal optimal strategies (3) are given by

$$\mathbf{u}^{(i)}(t, \mathbf{x}, \epsilon) = (-1)^i \mu \frac{\mathbf{x} - a^{(i)}}{||\mathbf{x} - a^{(i)}||},$$
$$\mathbf{v}^{(i)}(t, \mathbf{x}, \epsilon) = -(-1)^i \nu \frac{\mathbf{x} - a^{(i)}}{||\mathbf{x} - a^{(i)}}.$$

Let the following initial conditions and values of parameters be given: $t_0 = 0$, $\xi_0 = (0.9, 0.6)$, $\dot{\xi}_0 = (-1, -1)$, $\mu = 1.3$, $\nu = 0.7$, $\theta = 2.6$. Then we have $\mathbf{x}_0 = (-1.7, -2)$. Two variants of target points were considered:
(V1) $a^{(1)} = (5.5, -4)$, $a^{(2)} = (2.5, 3)$
(V2) $a^{(1)} = (7, -4)$, $a^{(2)} = (4, 4)$.
Time step for N-trajectory was 0.001.

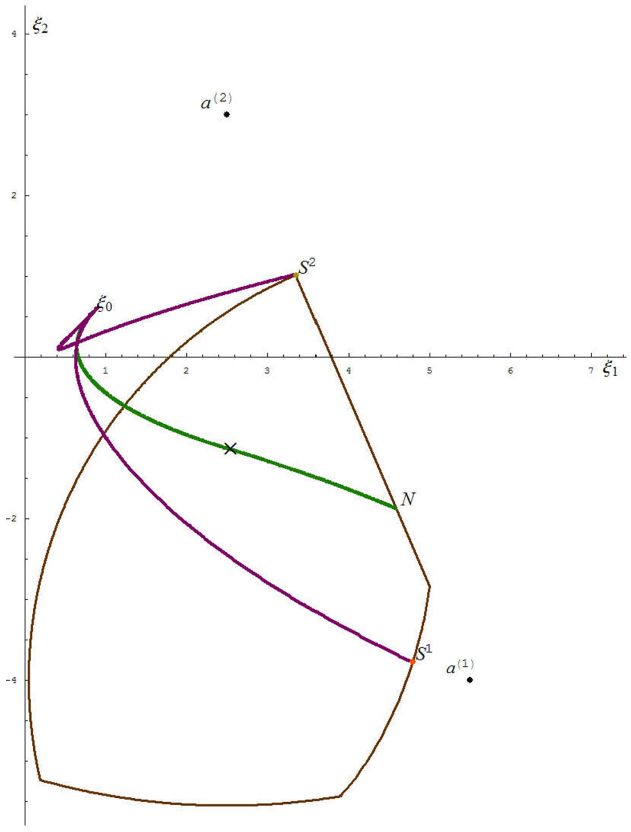

Fig. 1. V1: Optimal trajectories: N- and S-solutions

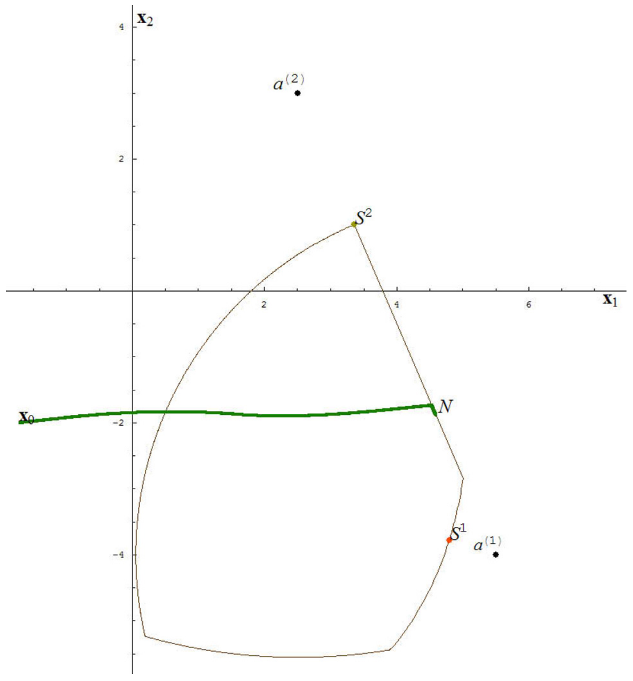

Fig. 2. V1: N-trajectory in reduced system

Fig. 1 and Fig. 2 show computed trajectories for variant V1. Fig. 1 depicts $S1$-, $S2$-trajectories and N-trajectory generated by BM-procedure in (ξ_1, ξ_2) plane. Symbols S^1 and S^2 on figures denote endpoints of $S1$ and $S2$ solution trajectories, respectively, while N denotes N-solution trajectory endpoint. On Fig. 1 symbol "$\times$" is used to show a position $(\xi_1(t^*), \xi_2(t^*))$ corresponding to moment $t^* = 1.882$. Starting from the moment, position of the game, calculated along the N-trajectory, changes its type from non-antagonistic to globally antagonistic. The fact of position type change is well illustrated at Fig. 2, where N-trajectory is shown in the plane (x_1, x_2). Since the time t^* the trajectory lies on line segment connecting $a^{(1)}$ and $a^{(2)}$ target points.

Fig. 3 shows computed trajectories for variant V2.

Corresponding controls (every 25th pair of vectors $(\mathbf{u}, \mathbf{v})$ of both players for Nash solution trajectories are shown on Fig. 4 for variant V1 and on Fig. 5 for variant V2. On Fig. 1, Fig. 3 and Fig. 2 also border of relevant $D = (D_1 \cap D_2) \setminus R$ set is painted, R denotes a half-plane bounded by a line connecting points $a^{(1)}$ and $a^{(2)}$ and not containing point ξ_0. Set D contains all N-trajectories endpoints, but in general case, there may be also points, which are not endpoints of any N-trajectories. D_i and S-trajectories were built with time step 0.005.

7. CONCLUSION

As it is seen today, there are three ways of development of the results presented. First, it seems to be possible to transparently generalize N- and S-solution algorithms for

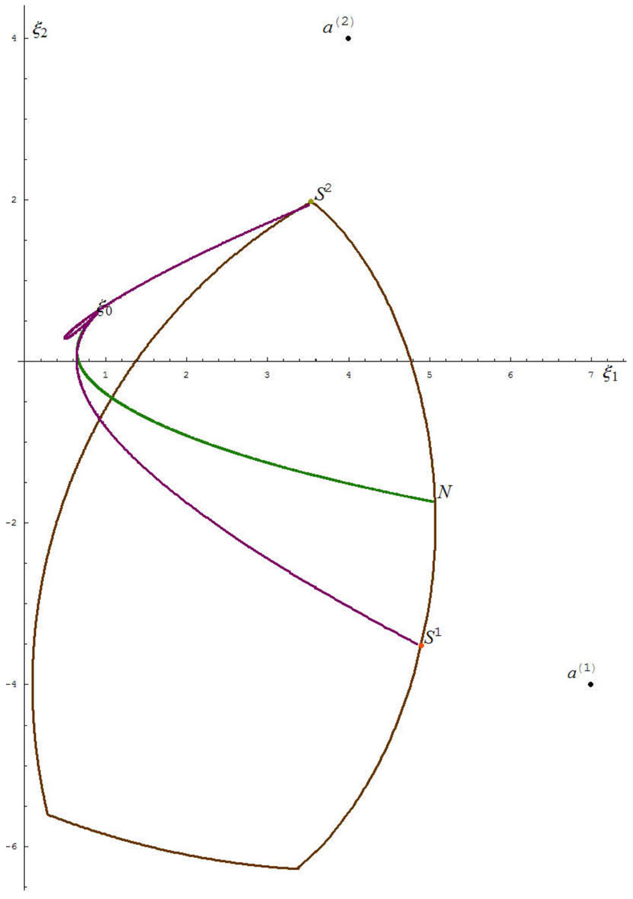

Fig. 3. V2: Optimal trajectories: N- and S-solutions

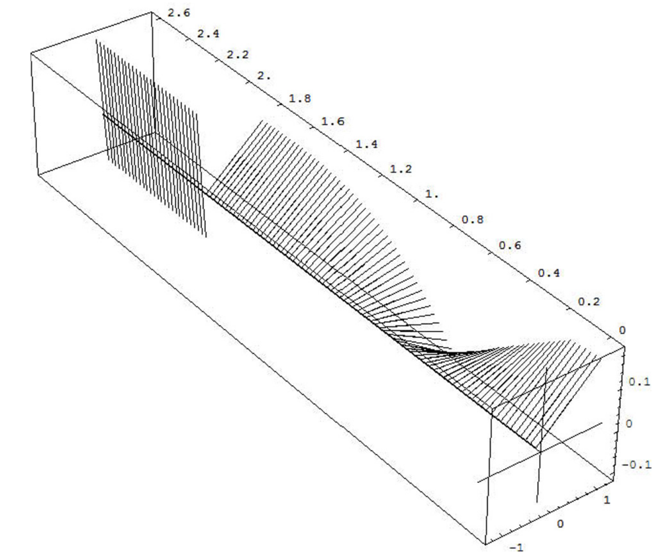

Fig. 4. V1: Player controls: N-solution

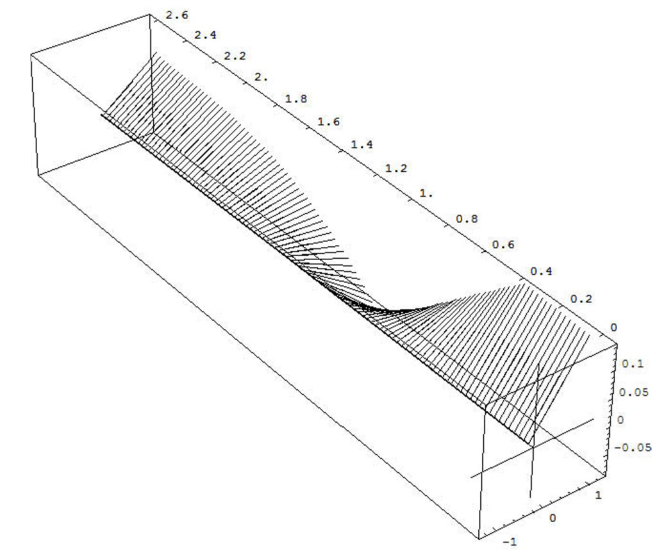

Fig. 5. V2: Player controls: N-solution

non-linear systems with dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{F}_1(t, \mathbf{x}(t), \mathbf{u}(t)) + \mathbf{F}_2(t, \mathbf{x}(t), \mathbf{v}(t)).$$

Second, software development may lead to a powerful and flexible framework simplifying solution computations in a class of differential games. The last program implementation uses techniques of generic programming, which is common for modern C++ software like CGAL. This supports flexibility of its structure and simplifies future modernizations. For example, the early experience allows to suggest, that facilities supplied by the library could give the algorithms literally new dimension: polygons could be changed to polyhedrons without deep change in generic algorithms constructing the solutions.

Finally, an algorithm approximating all N-trajectories endpoints set is planned. If this could be done, one may choose there an endpoint, which is optimal in some sense, and build N-solution, leading to that point, by back propagation similar to that used for S-solutions.

REFERENCES

Basar, T. and Olsder, G. (1982). *Dynamic Noncooperative Game Theory.* Acad. Press, N.Y.

Isakova, E., Logunova, G., and Patsko, V. (1984). Stable bridges construction in linear differential game with fixed final time. In A. Subbotin and V.P. and (eds.), *Algorithms and programs for linear differential games solutions*, 127–158. Ural Sci. Center of Acad. Sci. of USSR, Sverdlovsk. (In Russian).

Kleimenov, A. (1993). *Positional Differential Nonantagonistic Games.* Nauka, Urals Branch, Ekaterinburg. (In Russian).

Kleimenov, A. (1997). Solutions in a nonantagonistic positional differential game. *J. Appl. Naths Mechs*, 61(5), 717–723.

Kleimenov, A. and Osipov, S. (2003). Computation of stackelberg trajectories in a class of two-person linear differential games with terminal players' payoffs and polygonal constraining for controls. In *IFAC Workshop on Control Applications of Optimization, Preprints, Elsevier Science Ltd., Oxford.*, 201–205.

Kleimenov, A., Osipov, S., Cherepov, A., and Kuvshinov., D. (2006). A numerical solution for a hierarchical differential game of two persons. *Proc. of Ural State Univ.*, (46), 69–78. (In Russian).

Krasovskii, A. and Krasovskii, N. (1995). *Control under Lack of Information.* Birkhäuser, Berlin.

Krasovskii, N. (1985). *Control of a Dynamical System.* Nauka. (In Russian).

Krasovskii, N. and Subbotin, A. (1988). *Game-Theoretical Control Problems.* Springer-Verlag, NY. Berlin.

Vahrushev, V., Tarasiev, A., and Ushakov, V. (1987). An algorithm of union and intersection of sets in plane. In A. Subbotin and V.U. and (eds.), *Control with guaranteed result*, 28–36. Ural Sci. Center of Acad. Sci. of USSR, Sverdlovsk. (In Russian).