EPiC Series in Computing

Volume 114, 2026, Pages 152–171

Proceedings of the Workshop on Mathematical Modeling and Scientific Computing

Trajectory Simulation for Conflict-Controlled Systems with Multidimensional Control Sets

Anton V. Mikhailov[1,2]

[1] Krasovskii Institute of Mathematics and Mechanics, Russian Academy of Sciences, Yekaterinburg, Russia
tigr-mav@yandex.ru
[2] Ural Federal University, Yekaterinburg, Russia

Abstract

Zero-sum differential games are considered with a multi-dimensional phase vector, dynamics linear in the phase vector and players' controls, a fixed terminal instant, geometric constraints on the controls of the players, and a convex cylindrical terminal set. The constraints on the players' controls and the terminal set are multi-dimensional convex compacta in their respective spaces. Solving such a game includes constructing the maximal stable bridge and then simulating the trajectories of the system. Here the sections of the bridge are assumed to be constructed beforehand and given by their face lattices. Numerical procedures are proposed for simulating the trajectories of a conflict-controlled system under four strategies: the extremal strategy, constant control, random control, and an externally defined strategy. The strategies of the first and second players are prescribed independently, so an arbitrary pair of them may be used in a single simulation. The extremal strategy is implemented on the polytopal sections: the state is located by the hyperplane representation, and the aiming point is obtained by orthogonal projection when the state is interior and by a search over the face lattice when it is exterior. The random control is sampled as a point uniformly distributed on a face of a prescribed dimension of the constraint polytope, the polytope itself included. The procedures are illustrated by examples in which both the equivalent game and the control constraints are either two-dimensional or three-dimensional.

Keywords: Zero-sum linear differential games, Maximal stable bridge, Trajectory simulation, Extremal shift strategy, Convex polytope, Face lattice, Uniform sampling

1 Introduction

The paper is devoted to the numerical study of differential games [11, 15, 16, 20]. The class under consideration consists of differential games with a multi-dimensional phase vector, dynamics linear in the phase vector and players' controls, a fixed terminal instant, geometric constraints on the controls of the players, and a convex cylindrical terminal set.

The set of positions from which the first player can bring the state to the terminal set despite the action of the second one is the *solvability set* of the game. Its time sections are described by

the alternating integral of L.S. Pontryagin [20,21]. In the theory of positional differential games of N.N. Krasovskii and A.I. Subbotin, this set is a stable bridge. However, a more interesting object is a bridge maximal by inclusion called *maximal stable bridge*. The optimal control of the first player is realized by the strategy extremal to the bridge [15,16].

Thus, the maximal stable bridge is the main object to be constructed when a differential game with a target set is solved. Moreover, if a game with a terminal payoff is considered and the solution of the game is its value function, one can approximate the value function by a collection of its Lebesgue sets, each of which is constructed as a maximal stable bridge from a Lebesgue set of the payoff taken as the target set.

Maximal stable bridges can rarely be constructed analytically, so numerical methods are needed.

There are many grid-based methods operating on a discrete representation of the value function or of the solvability set, which are applicable to differential games of different classes including nonlinear problems and problems with state constraints [1,2,5,10,19]. However, grid methods are usually very demanding of memory and computational time.

Another approach to constructing maximal stable bridges is geometric. Its basis for the games of the above-mentioned class is provided by Pontryagin's second method [20]. It is a backward recursive procedure whose step involves two operations on convex sets: the *Minkowski sum* and *difference* (called *algebraic sum* and *geometric difference* in Russian literature). This procedure produces some approximations of time sections of the bridge on some time grid.

Geometric methods operate on the sections themselves. In 1984, Isakova, Logunova, and Patsko developed a complete computational procedure for the planar problem [12]. Zarkh and Ivanov then constructed a procedure in an arbitrary dimension of the phase space under the restriction that both control constraints are centrally symmetric parallelotopes: by associativity of the operations, every step is reduced to the Minkowski sum or difference of a convex polytope and a segment [23]. For a terminal set of an arbitrary dimension and for convex polytopal data of a general form, the problem was solved by Botkin and Ryazantseva [3]: they proposed a numerical algorithm of the backward step and implemented it. Their algorithm operates on the hyperplane representation of the polytopal sections of the bridge.

The author previously developed a procedure in which the input and the output of both operations of the recursion are multi-dimensional polytopes represented by their face lattices [17]. The face lattice records the incidences between the faces, which makes the algorithms of the simulation more direct than in the coarser representations. The present paper studies how this representation is used to simulate system trajectories based on the constructed sections.

In the linear-convex games of [14] the trajectories are simulated under several pairs of strategies combining the extremal shift to the accompanying points, an equiprobable random control, an externally prescribed greedy law, and the zero control, and the realized payoff is compared with the computed value of the game. In [9] the extremal shift is performed on a family of bridges, and the second player acts as a piecewise-constant random disturbance.

Trajectories are simulated under four strategies: the strategy extremal to the bridge, constant control, random control, and an externally defined strategy. The strategies of the two players are chosen independently.

The extremal strategy needs a point on the boundary of the section nearest to the current state. For a state outside the section, this is the classical nearest-point problem for a polytope, surveyed in the introduction of [8]. Those methods operate on the vertex or the hyperplane representation of the polytope. Here, the face lattice is available, which makes the algorithm simpler.

The distribution used for the random control is a mixture over the dimensions of the faces

of the constraint polytope: every dimension may carry positive mass, and the point is uniformly distributed on the selected face. The sample is produced by a descent over an auxiliary graph built from the face lattice. The procedure combines the pyramidal decomposition, the composition method for mixtures, and the Dirichlet distribution on a simplex.

The contribution of the paper is a simulation procedure for the class of games described, with the strategies of the players chosen independently. In this procedure, the extremal strategy is implemented on polytopal sections: the state is located by the hyperplane representation, the aiming point is obtained by orthogonal projection when the state is interior and by a search over the face lattice when it is exterior, and the computational cost of each step is estimated. The procedures have been implemented by the author in a program that uses the library for constructing the sections [17].

The rest of the paper is organized as follows. Section 2 states the original game and derives the equivalent game in the forecast coordinates. Section 3 recalls the construction of the sections of the maximal stable bridge and the face lattice of a polytope. Section 4 describes the numerical construction of trajectories and the four strategies. Section 5 presents two-and three-dimensional examples with the trajectories and the sections.

2 Problem formulation

2.1 Original game

A linear differential game is considered:

$$\begin{aligned} &\dot z = A(t)z + B(t)u + C(t)v, \\ &t \in [t_0, T], \quad z \in \mathbb{R}^n, \\ &u \in \mathbf{P} \subset \mathbb{R}^p, \quad v \in \mathbf{Q} \subset \mathbb{R}^q, \\ &z(T) \in \mathbf{M} + \mathit{MSpace}^{\perp}. \end{aligned} \tag{2.1}$$

The terminal instant T of the game is fixed. The controls u and v of the first and second players have geometric constraints $\mathbf{P}$ and $\mathbf{Q}$ on their instantaneous values, respectively. These sets are convex compacta in their respective spaces. The matrices $A(\cdot)$, $B(\cdot)$, and $C(\cdot)$ are continuous functions of time. The first player tries to guide the system to a cylindrical terminal target set at the terminal instant T, and the second player tries to deviate the system state from this set. The set has a convex compact base $\mathbf{M}$ which is located in the subspace *MSpace* spanned by d, $d \leqslant n$, selected coordinates with indices s_1, s_2, …, s_d. The base $\mathbf{M}$ is assumed to be of full dimension in *MSpace*.

2.2 Equivalent game

Consider the variable change

$$x(t) = X_{s_1,s_2,\ldots,s_d}(T,t)z(t) \tag{2.2}$$

defined by the matrix formed by d rows of the fundamental Cauchy matrix $X(T,t)$ for the system $\dot z = A(t)z$. The chosen rows correspond to the components of the phase vector that span the subspace *MSpace* containing the base $\mathbf{M}$ of the target set.

After such a change, one obtains an *equivalent differential game*:

$$\begin{aligned} &\dot x = D(t)u + E(t)v, \qquad t \in [t_0, T],\ x \in \mathbb{R}^d,\ u \in \mathbf{P},\ v \in \mathbf{Q},\ x(T) \in \mathbf{M}, \\ &D(t) = X_{s_1,s_2,\ldots,s_d}(T,t)B(t), \quad E(t) = X_{s_1,s_2,\ldots,s_d}(T,t)C(t). \end{aligned} \tag{2.3}$$

This new game (2.3) possesses the following properties:

- the dimension d of the new phase vector x is equal to the dimension of the set $\mathbf{M}$ defining the terminal target set in game (2.1);
- the right-hand side of dynamics (2.3) does not depend on the phase vector x;
- now, $\mathbf{M}$ itself is the target set.

The meaning of the new coordinates x is as follows. If the original system (2.1) is at a position $\big(t, z(t)\big)$, then under zero controls of the players (along the free motion of the system), the d phase coordinates defining the subspace of $\mathbf{M}$ reach the value $x(t)$ at the terminal instant T. These coordinates are therefore often called *forecast coordinates* or the *zero-effort miss*. The players' controls in game (2.3) affect the final state of the system.

Equivalence of the original and new coordinates z and x means the following. If, from a position $\big(t, z(t)\big)$ of game (2.1), the first player is able to guide the system to the target set at the terminal instant T, then in the equivalent game (2.3), the first player can also achieve its goal from the corresponding position $\big(t, x(t)\big)$. Conversely, if the first player in game (2.3) can guide the system to the target set at the instant T from the position $\big(t, x(t)\big)$, then in the original game (2.1), the first player can succeed from any position $\big(t, z(t)\big)$ such that $x(t)$ and $z(t)$ are connected by relation (2.2).

3 Construction of maximal stable bridge

3.1 Multi-dimensional polytopes

The suggested procedures deal with polytopal representations of all sets, so let us introduce some definitions and notation related to polytopes. A *polytope* is the convex hull of a finite set of points or, equivalently, a bounded intersection of finitely many closed half-spaces [24]. In contrast to a *polyhedron*, which may be unbounded, a polytope is bounded.

A *face* of a polytope K is its intersection with a hyperplane leaving the whole of K in one of the closed half-spaces it bounds. The empty set and K itself are faces as well. Any face itself is again a polytope, and its *dimension* is the dimension of its affine hull (the minimal affine subspace containing this face). The faces of dimension 0 are the *vertices* of K and those of dimension $\dim K - 1$ are its *facets*.

A face contained in another one is called its *subface*. The *face lattice* $\texttt{FLrep}\, K$ is the set of all faces of K with the partial order produced by inclusion of faces. It is graded by the dimension, so that the faces of one dimension form a *layer* of it. Figure 3.1 shows a three-dimensional cube and its face lattice. The vertices a and b and the edge ab they span are highlighted to relate the two panels.

There are three conventional representations of a polytope. The face lattice is one of them. It is the most informative representation and also the most demanding of computational resources, memory and processor time. A vertex representation $\texttt{Vrep}\, K$ (or V-representation) of a polytope K is the set of the vertices alone. The hyperplane representation $\texttt{Hrep}\, K$ (or H-representation) is a system of linear inequalities with one inequality per facet, whose solution set is K and whose normals are outward.

In the face lattice, the vertices form the layer above the empty face and the facets the layer below the polytope. Hence, the conversions $\texttt{FLrep}\, K \to \texttt{Vrep}\, K$ and $\texttt{FLrep}\, K \to \texttt{Hrep}\, K$ reduce to reading a single layer of the lattice.

3.2 Numerical procedure

For the sake of clarity of the part dealing with constructing optimal controls, let us pass from game (2.3) to *two* games: from the perspectives of the first and second players.

Both new games have the same dynamics and constraints for the players' controls. The difference is in the players' aims.

In the first game, consider the same target set $\mathbf{M}^{(1)} = \mathbf{M}$ which the first player aims to bring the system to and the second tries to avoid. In the second game, the target set is taken as $\mathbf{M}^{(2)} = \operatorname{cl}\big(\mathbb{R}^d \setminus \mathbf{M}\big)$. And in this game, the roles of the players are changed: the second player tries to bring the system to this set, and the first one hinders this.

Let $\mathbf{W}^{(1)}$ be the maximal stable bridge of the first player in the first game and $\mathbf{W}^{(2)}$ be the maximal stable bridge of the second player in the second game. Due to the Krasovskii – Subbotin alternative theorem [15, 16], these bridges are connected by the relation

$$\mathbf{W}^{(2)}(t) = \operatorname{cl}\big(\mathbb{R}^d \setminus \mathbf{W}^{(1)}(t)\big), \tag{3.1}$$

where $\mathbf{A}(t)$ denotes a time section of a set $\mathbf{A} \subset [t_0, T] \times \mathbb{R}^d$: $\mathbf{A}(t) = \big\{x \in \mathbb{R}^d : (t, x) \in \mathbf{A}\big\}$.

The bridge $\mathbf{W}^{(1)}$ can be used to construct the optimal control of the first player in game (2.3), and the bridge $\mathbf{W}^{(2)}$ is helpful for the second player's optimal control in the same game. The latter occurs due to relation (3.1) despite the fact that formally the bridge is constructed for another game. Also, due to this relation, one can construct only the bridge $\mathbf{W} = \mathbf{W}^{(1)}$ exactly or approximately.

Throughout the paper, a boldface letter denotes a true set connected with the game: the constraints $\mathbf{P}$ and $\mathbf{Q}$ for the players' controls, the terminal target set $\mathbf{M}$. The numerical procedures deal with some multi-dimensional polytopal approximations of these sets, which are denoted by the same italic letters P, Q, M. The produced polytopal approximations of time sections $\mathbf{W}(t)$ are denoted by $W(t)$.

In [20], L.S. Pontryagin proposed a theoretical procedure to construct a time section $\mathbf{W}(t_0)$ of a maximal stable bridge $\mathbf{W}$ at the instant t_0. In that work, the time section is called the *alternating integral*. The procedure itself was later named *Pontryagin's second method*.

Introduce a grid $\mathcal{T} = \{t_0 < t_1 < t_2 < \ldots < t_N = T\}$ in the time interval $[t_0, T]$ of the game. Denote $\Delta_i = t_{i+1} - t_i$, $i = 0, \ldots, N-1$; $\operatorname{diam}\mathcal{T} = \max \Delta_i$. The grid may be either uniform or non-uniform. This does not affect the idea of the procedure, so below it is assumed to be uniform with the step Δ.

For the instants t_i of the chosen grid $\mathcal{T}$, sets $W_i = W(t_i)$ approximating the time sec-

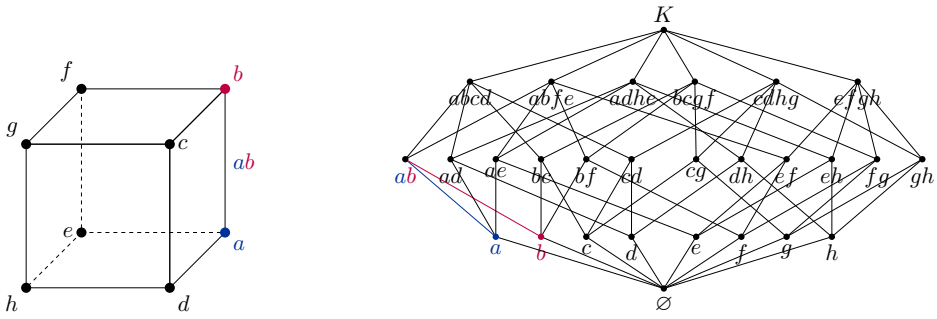


Figure 3.1: A three-dimensional cube K (left) and its face lattice $\texttt{FLrep}\,K$ (right).

tions $\mathbf{W}(t_i)$ are constructed by the procedure:

$$W_N = M, \quad W_i = \big(W_{i+1} + (-\Delta)\cdot D(t_i)P\big) \stackrel{*}{-} \Delta\cdot E(t_i)Q,\ i = N-1,\dots,0. \tag{3.2}$$

The symbols "$+$" and "$\stackrel{*}{-}$" denote the Minkowski sum and difference respectively (the algebraic sum and the geometric difference in the terminology of [21]):

$$A + B = \big\{a + b \in \mathbb{R}^d \;\big|\; a \in A, b \in B\big\}, \tag{3.3}$$

$$A \stackrel{*}{-} B = \big\{x \in \mathbb{R}^d \;\big|\; x + B \subset A\big\}. \tag{3.4}$$

Procedure (3.2) is a discrete scheme, and the set it produces depends on the time grid $\mathcal{T}$ and on the sets M, P, Q used in place of the true sets of the game. Convergence with respect to both the grid and the sets holds in the following sense. Consider a sequence of procedures (3.2) with time grids $\mathcal{T}_n$ and approximations M_n, P_n, Q_n such that

$$\operatorname{diam}\mathcal{T}_n \to 0, \quad h(M_n, \mathbf{M}) \to 0, \quad h(P_n, \mathbf{P}) \to 0, \quad h(Q_n, \mathbf{Q}) \to 0$$

as $n \to \infty$, where $h(\cdot,\cdot)$ denotes the Hausdorff distance between compacta. Let $W_0^{(n)}$ be the set W_0 built by the procedure with the grid $\mathcal{T}_n$ and the approximations M_n, P_n, Q_n. Then $h\big(W_0^{(n)}, \mathbf{W}(t_0)\big) \to 0$ as $n \to \infty$, as follows from [13, 18]. The sets W_i produced by procedure (3.2) are assumed throughout to contain a ball of a fixed radius, so they are full-dimensional with non-empty interior in $\mathbb{R}^d$. Convergence of the scheme in the case when it deals with true sets $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{M}$ and the only cause of inaccuracies is the time discretization is proven by L.S. Pontryagin himself in [21].

Here, the approximations M, P, and Q are chosen as polytopes in their spaces. The recursive scheme (3.2) then yields polytopes W_i. Polytopal data make the procedure well suited to computer implementation: one needs algorithms for the Minkowski sum (3.3) and difference (3.4).

The algorithm for the Minkowski sum is taken by the author from [6] and deals with face lattices of the summands. Therefore, one needs to construct the face lattice of the first player's *vectogram* $D(t_i)P$ (that is, the set of all possible velocities this player can apply to the system). This can be done by the procedure for constructing the convex hull of a finite set of points, which is taken by the author from a classical paper [22].

The idea of the Minkowski difference procedure is taken from [3]. This procedure needs the H-representation of the minuend and a point representation of the subtrahend. The point representation can even be a *redundant V-representation* $\overset{*}{\texttt{Vrep}}$. One needs only a swarm of points including not only vertices of the polytope, but also any of its internal points (that is, a collection of points whose convex hull is the necessary polytope). Thus, there is no need to construct the face lattice $\texttt{FLrep}\big(E(t_i)Q\big)$ of the second player's vectogram.

However, the difference, which is the new time section of the bridge, is produced as a *redundant H-representation* $\overset{*}{\texttt{Hrep}}$, that is, a collection of linear inequalities, some of which can be excluded without changing the resultant polytope. It should be reduced to a non-redundant H-representation and converted to the face lattice to be used in the Minkowski sum at the next step of procedure (3.2). The author suggested a procedure that constructs a vertex representation on the basis of a redundant H-representation.

The operations necessary for procedure (3.2): $\operatorname{conv}\{\text{points}\} \to \texttt{FLrep}$; $\texttt{Hrep} \to \texttt{FLrep}$; $\texttt{FLrep} \to \texttt{Vrep}$; $\texttt{FLrep} \to \texttt{Hrep}$; $\texttt{FLrep} + \texttt{FLrep} \to \texttt{FLrep}$; $\texttt{Hrep} \stackrel{*}{-} \overset{*}{\texttt{Vrep}} \to \overset{*}{\texttt{Hrep}}$; $\overset{*}{\texttt{Hrep}} \to \texttt{Vrep}$ have already been implemented by the author [17] in a computer library written in C# for the .NET application platform.

Since the precision of computations is essential for geometric algorithms, the procedures in the library are parametrized by the numeric type. Any type with the standard arithmetic and elementary functions may be substituted. Thus, the same computation runs in standard or extended precision without any change to the code. Also, all comparison operations are performed with some tolerance common to the entire library.

4 Trajectory simulation

4.1 Numerical trajectory construction

The problem considered here is the simulation of trajectories of system (2.1) under various strategies of the players. To compute such a trajectory, the controls of both players have to be specified at every instant of the simulation.

A *strategy* is a rule that returns a control value on the basis of the information available at the current instant.

A trajectory is computed on some time grid, which, generally speaking, can differ from the grid $\mathcal{T}$ of the bridge construction introduced in Section 3. The trajectory emanates from a given initial state $z_0 \in \mathbb{R}^n$. At every grid instant t_i, the strategies chosen for the first and second players are evaluated and return the controls u_i and v_i. After both controls have been generated, the state is advanced to the next grid instant by the explicit Euler scheme:

$$z_{i+1} = z_i + \Delta \cdot \big(A(t_i)z_i + B(t_i)u_i + C(t_i)v_i\big), \qquad i = 0, \ldots, N-1.$$

The program implements four strategies: the strategy of extremal shift (which gives the optimal control), constant control, random control, and an externally defined strategy. The procedures described in this section are implemented in a program that uses the library outlined in Section 3. For each simulation, the strategies of the two players can be chosen independently of each other, so an arbitrary pair of them may be used in a single simulation.

4.2 Extremal strategy

This strategy is introduced by N.N. Krasovskii and A.I. Subbotin [15, 16]. The idea of this strategy is as follows: when the position of the system is inside a stable bridge, the player can apply any control; if the system is outside the bridge, then the control should be chosen in such a way as to steer the system towards the bridge as much as possible.

In the latter case, one should find the point of the time section of the player's bridge at the current instant closest to the current position of the system. It is called the *aiming point*. The vector from the current position to the aiming point is the steering direction. It is called the *aiming vector*. The extremal shift control is such a value which maximizes the projection of the system velocity along the aiming vector.

Since the computations from Section 3 give bridges in the space of the equivalent game (2.3), the extremal strategy works in the space $\mathbb{R}^d$ of this game. Let z_i be the current state of the original system (2.1). The state in the forecast coordinates is obtained by the transformation (2.2):

$$x_i = X_{s_1,s_2,\ldots,s_d}(T, t_i)z_i.$$

For an instant t_i from the grid where the trajectory is computed, let t_{i^*} be the instant of the bridge construction grid closest to t_i. The strategy uses the time section $W(t_{i^*})$ of the maximal stable bridge W.

By relation (3.1), the section of the first player's bridge is $W(t_{i^*})$ and that of the second player's bridge is $\operatorname{cl}\big(\mathbb{R}^d \setminus W(t_{i^*})\big)$. For $k = 1, 2$, the aiming point $b^{(k)}$ of the kth player is the point of the section of that player's bridge nearest to the state x_i, and the aiming vector of that player is $l^{(k)} = b^{(k)} - x_i$. If the state belongs to the section of the player's bridge, then $b^{(k)} = x_i$ and the aiming vector is zero. Otherwise, the aiming point lies on the boundary $\partial W(t_{i^*})$, which is common to the sections of both bridges.

If $x_i \notin W(t_{i^*})$, the state is outside the section of the first player's bridge, and that player aims at this section, choosing a control that is extremal in the direction $l^{(1)}$:

$$u \in \operatorname*{Arg\,max}_{u' \in \mathtt{Vrep}\, P} \langle D(t_i)u', l^{(1)} \rangle. \tag{4.1}$$

Note that the maximization is performed not over all points of the polytopal approximation P of the control constraint $\mathbf{P}$, but over its vertices only. This is legitimate because the maximized function is linear in u', and a linear function attains its maximum on a polytope at a vertex [24]. If the set of maximizers is not a single vertex, every point of it is extremal in this direction as well, and the strategy returns a vertex among them. Wherever the strategy prescribes a control, that control is therefore a vertex of the constraint polytope.

If $x_i \in \operatorname{int} W(t_{i^*})$, the state is outside the section of the second player's bridge, and that player aims from the interior of the section towards its boundary:

$$v \in \operatorname*{Arg\,max}_{v' \in \mathtt{Vrep}\, Q} \langle E(t_i)v', l^{(2)} \rangle. \tag{4.2}$$

Again, the maximization is performed over vertices of Q only.

Finally, if $x_i \in \partial W(t_{i^*})$, the state is located on the boundary shared by the two sections: the aiming points coincide with it, $b^{(1)} = b^{(2)} = x_i$, both aiming vectors are zero, and the strategy prescribes no particular control to either player. Where a control is not prescribed, it may be any point of the constraint of the player. The program takes the point of the constraint polytope with the minimal norm. Unlike the prescribed controls, this point need not be a vertex.

The maximization amounts to one pass over the vertices. When the maximized function is rewritten as $\langle D(t_i)u', l^{(1)} \rangle = \langle u', D^\top(t_i)l^{(1)} \rangle$, the direction $D^\top(t_i)l^{(1)} \in \mathbb{R}^p$ is computed once in $O(dp)$ operations, and every vertex then costs one inner product in $\mathbb{R}^p$, which gives $O\big(dp + |\,\mathtt{Vrep}\, P|p\big)$ operations. For the second player, q and $\mathtt{Vrep}\, Q$ stand in place of p and $\mathtt{Vrep}\, P$.

4.2.1 Point location with respect to polytope

The extremal strategy needs to locate the state with respect to the section, that is, to distinguish the three cases above. The sections are produced by procedure (3.2) as face lattices $\mathtt{FLrep}\, W$, from which the hyperplane representation $\mathtt{Hrep}\, W$ is available:

$$W = \big\{x \in \mathbb{R}^d \;\big|\; g_j(x) \leqslant 0,\ j = 1, \ldots, |\,\mathtt{Hrep}\, W|\big\}, \quad g_j(x) = \langle n_j, x \rangle - c_j, \quad \|n_j\| = 1. \tag{4.3}$$

The normals n_j are outward and normalized, so $g_j(x)$ is the signed distance from x to the jth supporting hyperplane. Define

$$g^* = \max_j g_j(x_i), \qquad j^* \in \operatorname*{Arg\,max}_j g_j(x_i). \tag{4.4}$$

If the maximum is attained for several hyperplanes, the index of any of them may be taken as j^*. The point x_i is located strictly inside W if $g^* < 0$, on the boundary if $g^* = 0$, and

outside if $g^* > 0$. The two non-degenerate cases are shown in Fig. 4.1. For an interior point the j^*th hyperplane is one of those nearest to it, and it is this hyperplane that the computation of the aiming point uses. The maximum in (4.4) is found in one pass over the supporting hyperplanes with an inner product at every step, which gives a computational cost of $O(|\,\mathtt{Hrep}\,W|d)$ operations.



(a) The state outside the section of the first player's bridge.

(b) The state outside the section of the second player's bridge.

Figure 4.1: Localization of the state with respect to a section of the bridge.

4.2.2 Computing aiming point

The extremal strategy needs the aiming point of the player whose bridge section does not contain the state. This is the point of the boundary of the section nearest to the state. It is obtained differently in each of the three cases, and the cases are distinguished by the sign of the value g^* of localization (4.4), which also yields the index j^* attaining the maximum.

If $g^* = 0$, the state is located on the boundary and the aiming points of both players coincide with it, $b^{(1)} = b^{(2)} = x_i$.

If $g^* < 0$, the state is located strictly inside the polytope, so the aiming point to be computed is the one of the second player. It is obtained by orthogonal projection onto the j^*th supporting hyperplane, one of those nearest to the state:

$$b^{(2)} = x_i - g^* n_{j^*}.$$

Indeed, $g_j(x_i) \leqslant g^*$ by the definition of the maximum and $\langle n_j, n_{j^*}\rangle \leqslant 1$ because the normals are non-codirected unit vectors, so $g_j(b^{(2)}) = g_j(x_i) - g^*\langle n_j, n_{j^*}\rangle \leqslant g^*\big(1 - \langle n_j, n_{j^*}\rangle\big) \leqslant 0$ for every j. Hence, the point $b^{(2)}$ belongs to $W(t_{i^*})$, and $g_{j^*}(b^{(2)}) = 0$ places it on the boundary. No point of the boundary is closer to x_i because every such point is located on some supporting hyperplane and the distance from x_i to that hyperplane is at least $-g^*$.

If $g^* > 0$, the state is located outside the polytope, and the aiming point to be computed is the one of the first player. Here, no explicit formula is available, and the nearest boundary point $b^{(1)}$ is found by a search over the face lattice. A facet of $W(t_{i^*})$ is called *visible* from x_i if $g_j(x_i) > 0$ for its supporting hyperplane [24, Section 8.2]. Only the facets visible from x_i and their subfaces of all dimensions are examined, since the nearest boundary point belongs to a visible facet.

Every non-empty face G of the lattice is equipped with an affine basis: a point $o_G \in G$ and an orthonormal basis $e_1, \ldots, e_{d_G}$ of the directional subspace of $\operatorname{aff} G$, where $d_G = \dim G$. The

projection of the state onto this affine hull is

$$\pi_G(x_i) = o_G + \sum_{m=1}^{d_G} \langle e_m, x_i - o_G \rangle \, e_m. \tag{4.5}$$

This point belongs to $\operatorname{aff} G$, but not necessarily to the face G, so it is accepted as a candidate only if $g_j(\pi_G(x_i)) \leqslant 0$ for every j, that is, if it belongs to $W(t_{i^*})$. Among the candidates, the one nearest to x_i is returned. The search does find the nearest point: it is located in the relative interior of some face — its interior within the affine hull of that face — and the projection onto that affine hull is exactly this point.

When the system state is inside the time section $W(t_{i^*})$, the complexity of the procedure is $O(|\,\mathtt{Hrep}\,W| \cdot d)$ since the main operation is the computation of g^* and j^*. When the system is outside $W(t_{i^*})$, the complexity of the procedure is

$$O\big(|\,\mathtt{FLrep}\,W| \cdot (d^2 + |\,\mathtt{Hrep}\,W| \cdot d)\big) = O\big(|\,\mathtt{FLrep}\,W| \cdot |\,\mathtt{Hrep}\,W| \cdot d\big)$$

since in the worst case one needs to check (almost) every face in the face lattice (the factor $|\,\mathtt{FLrep}\,W|$) to project x_i onto the affine hull of the face (the term d^2), and to check if the projection belongs to the polytope of the time section (the term $|\,\mathtt{Hrep}\,W| \cdot d$). Since $|\,\mathtt{Hrep}\,W| \geqslant d$, the equality holds. With that, the affine bases of all faces are precomputed and stored in the face lattice.

4.3 Constant control

For either player, a constant control is specified by choosing a point of the control space and keeping it fixed throughout the game interval. Note that this point is not required to belong to the control constraint.

4.4 Random control

To simulate a player that acts without any purpose, its control is sampled at random from its constraint polytope. Several sampling scenarios are of interest. One of them produces a random point distributed uniformly in the polytope (a silly player). However, in this case, the point is inside the polytope with probability 1. Another approach generates a point distributed uniformly on the boundary of the polytope (a silly but strong player). In this case, the point hits the interior of a facet with probability 1. Hence, one more scenario is needed. It first chooses a dimension and then samples a point uniformly from the union of all faces of that dimension (in particular, a random vertex or a point on a random 1-dimensional edge). More generally, the dimension is chosen at random with prescribed probabilities (for example, only 0 and 1 to obtain a point on an edge or at a vertex).

Whatever the scenario, the core operation of the procedure is the same: to sample a point uniformly from a given face of the polytope (recall that the polytope itself is a face).

Denote the polytope under consideration by K and its dimension by d_K. Let its face lattice $\mathtt{FLrep}\,K$ be given (possibly, as a sublattice of the entire polytope). A face of dimension k is called a *k-face*. The 1-faces are the *edges*. Every face of K is itself a polytope, and its faces are faces of K. Below, F denotes a non-empty face of K and $k = \dim F$. $\operatorname{Vol}(F)$ denotes the k-dimensional volume of F. The volume of a vertex is set equal to one. A k-dimensional *simplex* is the convex hull of $k+1$ affinely independent points. In particular, every edge is a simplex.

Using `FLrep` K, the procedure *a priori* builds an auxiliary sampling structure, the *descent graph* described below. It is built once for the polytope.

The volumes of the faces are computed layer by layer, from the vertices up to K itself, in such a way that when a face is reached, the volumes of its facets are already known. If F has exactly $k+1$ vertices $w_0, \ldots, w_k$, it is a simplex. Its volume is given by the Gram determinant:

$$\mathrm{Vol}(F) = \frac{1}{k!}\sqrt{\det\left(VV^\top\right)}, \tag{4.6}$$

where the rows of the matrix V are the vectors $w_i - w_0$, $i = 1, \ldots, k$. For an edge, this is its length. If F is not a simplex, then $k \geqslant 2$. A vertex a_F of F is fixed, say, the first one in the list of its vertices. It is called the *descent vertex* of F. The face F is decomposed into pyramids with the apex a_F:

$$F = \bigcup_G \mathrm{conv}\left(\{a_F\} \cup G\right), \qquad \mathrm{Vol}\left(\mathrm{conv}\left(\{a_F\} \cup G\right)\right) = \frac{h_{a_F}(G)}{k}\ \mathrm{Vol}(G). \tag{4.7}$$

Here, the union is taken over the facets G of F that do not contain a_F. The value $h_{a_F}(G)$ is the distance from a_F to the affine hull $\mathrm{aff}\, G$ of the facet G. These pyramids have pairwise disjoint relative interiors and cover F, so the volume of F is the sum of their volumes. Every face is processed once, even if it is the base of pyramids of several faces of higher dimension. Its volume is reused in all of them.

The decomposition defines a directed acyclic graph on the non-empty faces of K, which is called the *descent graph*. The node of a face F that is not a simplex stores the descent vertex a_F and an arc $F \to G$ to every facet G of F not containing a_F. The weight $\omega(F, G)$ of this arc is the volume of the corresponding pyramid:

$$\omega(F, G) = \frac{h_{a_F}(G)}{k}\ \mathrm{Vol}(G), \qquad \mathrm{Vol}(F) = \sum_G \omega(F, G).$$

The node of a simplex face stores its vertices and its volume (4.6). It has no outgoing arcs.

A point is sampled uniformly from a simplex with vertices $w_0, \ldots, w_k$ as follows. Draw k independent values from the uniform distribution on the interval $(0, 1)$ and order them increasingly: $\xi_1 \leqslant \ldots \leqslant \xi_k$. Adjoining the ends of the interval, $\xi_0 = 0$ and $\xi_{k+1} = 1$, define

$$\lambda_i = \xi_{i+1} - \xi_i, \qquad i = 0,\ \ldots,\ k.$$

These numbers are non-negative and sum to one. Their joint distribution is the Dirichlet distribution with all parameters equal to one. The point

$$X = \lambda_0 w_0 + \ldots + \lambda_k w_k$$

is uniformly distributed in the simplex [7]. For $k = 0$, the single vertex is returned. The complexity of the procedure is $O(k \log k)$ (since its main part is ordering the values) plus the cost of forming the point.

To sample a point uniformly from a face F, the procedure starts from the node of F in the descent graph. If F is a simplex, the point is sampled as it is said above. Otherwise, the following steps are performed:

1. an outgoing arc $F \to G$ is chosen with probability $\omega(F, G)/\,\mathrm{Vol}(F)$;
2. a point Y is sampled uniformly from G by the same procedure applied recursively;

3. a value U is drawn from the uniform distribution on (0, 1) independently of the chosen arc and of Y;

4. the point $X = a_F + U^{1/k} \cdot (Y - a_F)$ is returned.

Each recursive step lowers the dimension by one, and the recursion stops at a simplex face, at an edge, at least, so it takes at most $k - 1$ steps.

All the scenarios use the same descent graph and differ only in the starting face. For a point in a prescribed face, the descent starts from this face. In particular, the descent from K gives a point uniformly distributed in the polytope. For the general scenario, the user prescribes a mixture over the dimensions of the faces:

$$\mu_0 + \mu_1 + \ldots + \mu_{d_K} = 1, \qquad \mu_k \geqslant 0.$$

First, a dimension k is chosen with probability μ_k. Then a k-face is chosen with probability proportional to its volume. Finally, the descent starts from this face. The choice $\mu_{d_K-1} = 1$ gives a point uniformly distributed on the boundary of K. The choice $\mu_0 = 1$ gives a vertex chosen with equal probabilities, since every vertex has unit volume.

The procedure described above is proposed by the author. It combines several known techniques: the pyramidal decomposition of a polytope, the composition method for sampling from a mixture of distributions, and the Dirichlet distribution on a simplex [7]. Bruns and Ichim [4] use the descent in the face lattice to compute the volume of a polytope. Here, a similar structure is used for sampling. The classical approach partitions a polytope into simplices and chooses a simplex with probability proportional to its volume [7]. The procedure above needs no such partition. Every arc of its descent graph corresponds to a pyramid over a facet of a face, so the graph is no larger than the face lattice $\texttt{FLrep}\,K$. A single descent graph serves sampling from any non-empty face of K and from any prescribed mixture over the dimensions of the faces.

The point returned is uniformly distributed in the selected face, and points returned by successive calls are independent. A proof of these properties and a full analysis of the computational cost are the subject of a separate work of the author, which is in preparation.

4.5 Externally defined strategy

A strategy may also be supplied by an external routine written by the user. At its first use, the source text of the routine is compiled into the running program and is simultaneously cached as a dynamic-link library (DLL). Subsequent runs load the routine from this library. At every grid instant, the routine receives all information about the game and its current state and returns a control value. The value returned need not belong to the constraint. The routine is created once per trajectory, so it may keep data between the instants of the grid.

An example of such a strategy is

$$u_i \in \operatorname*{Arg\,max}_{u' \in \texttt{Vrep}\,P} \langle \psi(t_i, z_i), u' \rangle, \tag{4.8}$$

where the steering direction $\psi(t_i, z_i) \in \mathbb{R}^p$ is prescribed by the user. The control returned is always a vertex of P. The maximization in the extremal strategy (4.1) has the same form with $\psi = D^\top(t_i) l^{(1)}$, the difference being that there the direction is determined by the stable bridge rather than supplied by the user.

5 Numerical examples

Two examples are considered below. In both of them, the system consists of m independent one-dimensional harmonic oscillators coupled through the controls of the players. The phase vector is $z = (r, p) \in \mathbb{R}^{2m}$, so that $n = 2m$: its first m components r are the coordinates of the oscillators and the last m components p are their velocities. The controls of both players are m-dimensional:

$$\dot{z} = \begin{pmatrix} 0 & \nu I_m \\ -\nu I_m & 0 \end{pmatrix} z + \begin{pmatrix} I_m \\ 0 \end{pmatrix} u + \begin{pmatrix} 0 \\ I_m \end{pmatrix} v, \qquad \nu = 0.5, \quad t \in [0, 4\pi]. \tag{5.1}$$

The first player acts on the coordinates, and the second one on the velocities. The coordinates r are the selected ones, so that $MSpace = \operatorname{span}(e_1, \ldots, e_m)$ and the equivalent game is m-dimensional, $d = m$.

For system (5.1), the Cauchy matrix and the matrices of the equivalent game are

$$X(T,t) = \begin{pmatrix} \cos \nu\tau\, I_m & \sin \nu\tau\, I_m \\ -\sin \nu\tau\, I_m & \cos \nu\tau\, I_m \end{pmatrix}, \qquad D(t) = \cos \nu\tau\, I_m, \qquad E(t) = \sin \nu\tau\, I_m, \tag{5.2}$$

where $\tau = T - t$.

In both examples the constraints of the players and the terminal set are themselves polytopes, so that $P = \mathbf{P}$, $Q = \mathbf{Q}$, $M = \mathbf{M}$. No approximation of the data of the game is involved, and the only error of the constructed sections is that of the time discretization. The sections are constructed on a uniform grid of $N = 128$ steps, $\Delta = 4\pi/128 \approx 0.098$, and the trajectories are computed on the same grid. The sections are computed by the library of Section 3, and the trajectories by the program of Section 4. Where the extremal strategy prescribes no control, the point of the constraint polytope nearest to the origin is taken. The constraints are centred at the origin, so this control is zero.

In both examples, the random control is sampled anew at every instant of the grid. From the standpoint of continuous-time control systems, such a rule for generating the control is not physically reasonable. The primary purpose of these examples, however, is to demonstrate the proposed procedure for sampling a point from a polytope. A more natural alternative would be to consider piecewise constant random controls with sufficiently long intervals over which the control remains constant.

5.1 Pair of oscillators

Here, $m = 2$. All three sets of the game are regular hexagons centred at the origin. The constraints $\mathbf{P}$ and $\mathbf{Q}$ of the players have circumradius 1.5 and are rotated by 30° with respect to each other, a vertex of $\mathbf{Q}$ located on the first axis. The terminal set $\mathbf{M}$ has circumradius 3 and is rotated by 15° with respect to $\mathbf{Q}$.

The trajectories are computed from the state $z_0 = (1.2403,\ 0.3101,\ 0.5,\ -0.5)$, whose forecast coordinates are located inside the section $W(t_0)$ close to its boundary. The extremal strategy of one player is opposed by four strategies of the other: the extremal strategy, shown by a solid blue line; the constant control, shown by a dash-dotted red line, at a vertex of the constraint — $(1.5,\ 0)$ for the second player and $1.5\,(\cos 30°,\ \sin 30°)$ for the first; random control, shown by a dashed orange line, uniformly distributed in the constraint, which is the choice $\mu_m = 1$ of the mixture of Subsection 4.4; and the externally defined strategy (4.8), shown by a dotted violet line, with the direction $\psi = (-r_2, r_1)$, that is, tangent to the circle about the origin. The sections are drawn in green. The same four pairs are then simulated with the roles of the players exchanged.

Each panel shows the section at the corresponding instant, whereas each curve extends from the initial instant to that instant. The lower-left corner of each panel shows the coordinate basis. Its arrows have unit length on the scale of the panel. The arrowhead of each curve marks the current state on the corresponding trajectory. The first panel of each series corresponds to the initial instant and shows the section $W(t_0)$ and the state $x_0 = x(t_0)$. The four intermediate panels correspond to the successive extrema of the size of the section: the section contracts almost to a point at $t = 1.57$, expands to a maximum at $t = 4.91$, contracts again at $t = 7.76$, and expands at $t = 11.09$. The final panel shows the terminal instant, at which the section is the terminal set M itself.

In Figure 5.1, where the first player uses the extremal strategy, the panel at the terminal instant shows the discreteness of the scheme: the state corresponding to the random control ends deep inside the terminal set, and the other three states are located near its boundary, some of them slightly outside it. The guarantee of the extremal strategy holds for the limit motions of the step-by-step scheme but not for the Euler polyline on a fixed grid. As the grid is refined, the distance from each terminal state to the terminal set tends to zero.

In Figure 5.2, where the second player uses the extremal strategy, that player achieves its goal against all four strategies of the first.

5.2 Triple of oscillators

Here, $m = 3$. All three sets of the game are regular octahedra centred at the origin, with their vertices on the coordinate axes. The constraints $\mathbf{P}$ and $\mathbf{Q}$ of the players coincide and have circumradius 1.5. The terminal set $\mathbf{M}$ is the octahedron of circumradius 1.7.

The trajectories are computed from the state $z_0 = (0.7,\ 0.4,\ -0.3,\ 0.6,\ -0.8,\ 1.1)$, whose forecast coordinates are located inside the section $W(t_0)$. As in the previous example, the extremal strategy of one player is opposed by four strategies of the other: the extremal strategy, shown in blue; the constant control, shown in red, at the vertex $(1.5,\ 0,\ 0)$ of the constraint; random control, shown in orange, on the boundary of the constraint, which is the choice $\mu_{m-1} = 1$ of the mixture of Subsection 4.4; and the externally defined strategy (4.8), shown in violet, with the direction $\psi = w \times r$, where w is the fixed unit vector along $(1, 2, 3)$. The direction w is deliberately in general position with respect to the octahedron: it is parallel neither to a vertex vector nor to a facet normal, so the sequence of the vertices the strategy visits is not degenerately symmetric. The same four pairs are then simulated with the roles of the players exchanged.

The conventions of the panels are those of the previous example. The coordinate basis is now three-dimensional. The four intermediate panels correspond to the successive extrema of the size of the section: the section contracts at $t = 1.57$, expands to a maximum at $t = 4.71$, contracts again at $t = 7.85$, and expands at $t = 11.00$. The final panel shows the terminal instant, at which the section is the terminal set M itself.

In Figure 5.3, where the first player uses the extremal strategy, all four trajectories remain near the sections throughout the game. The trajectory generated by the pair of extremal strategies follows the changing section, staying close to its boundary. At the terminal instant the state corresponding to the random control is deep inside the terminal set, whereas the other three are located outside it but close to its boundary, with the same reservation on the discreteness of the scheme as in the previous example.

In Figure 5.4, where the second player uses the extremal strategy, the state under the constant control moves far from the sections along an almost straight line and returns approximately along it. At the terminal instant it is near the boundary of the terminal set. For the

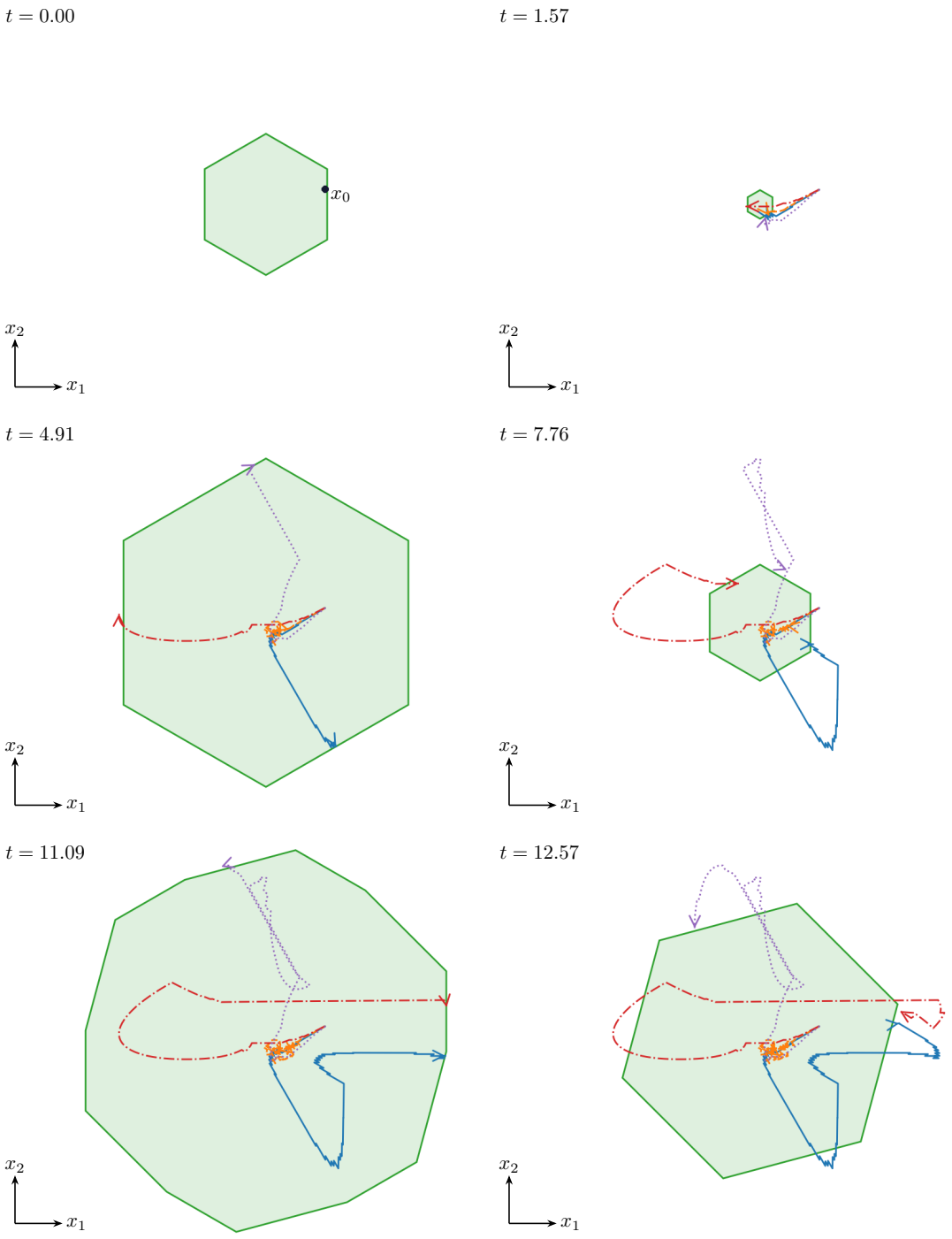


Figure 5.1: The first player's extremal strategy against the four strategies of the second.

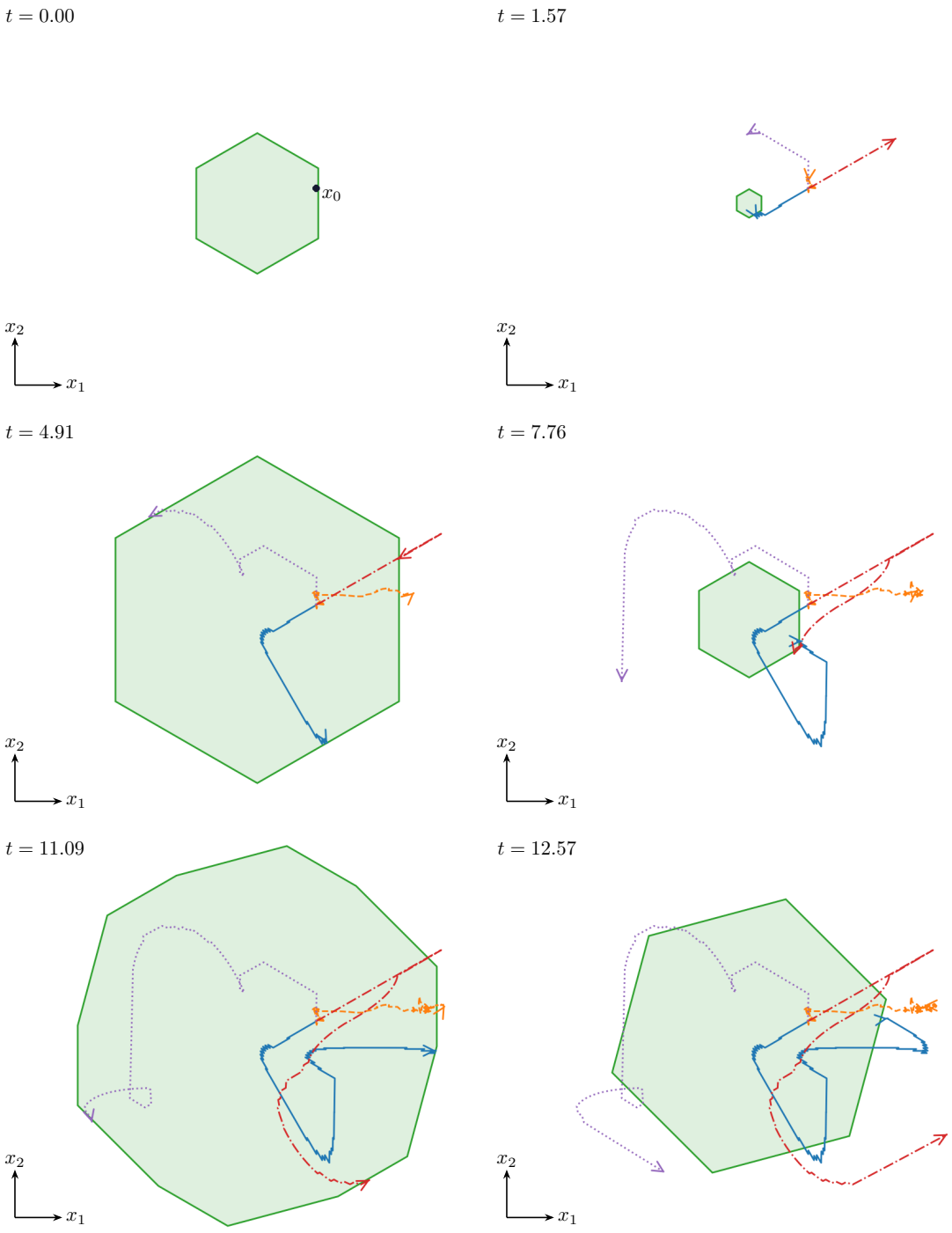


Figure 5.2: The second player's extremal strategy against the four strategies of the first.

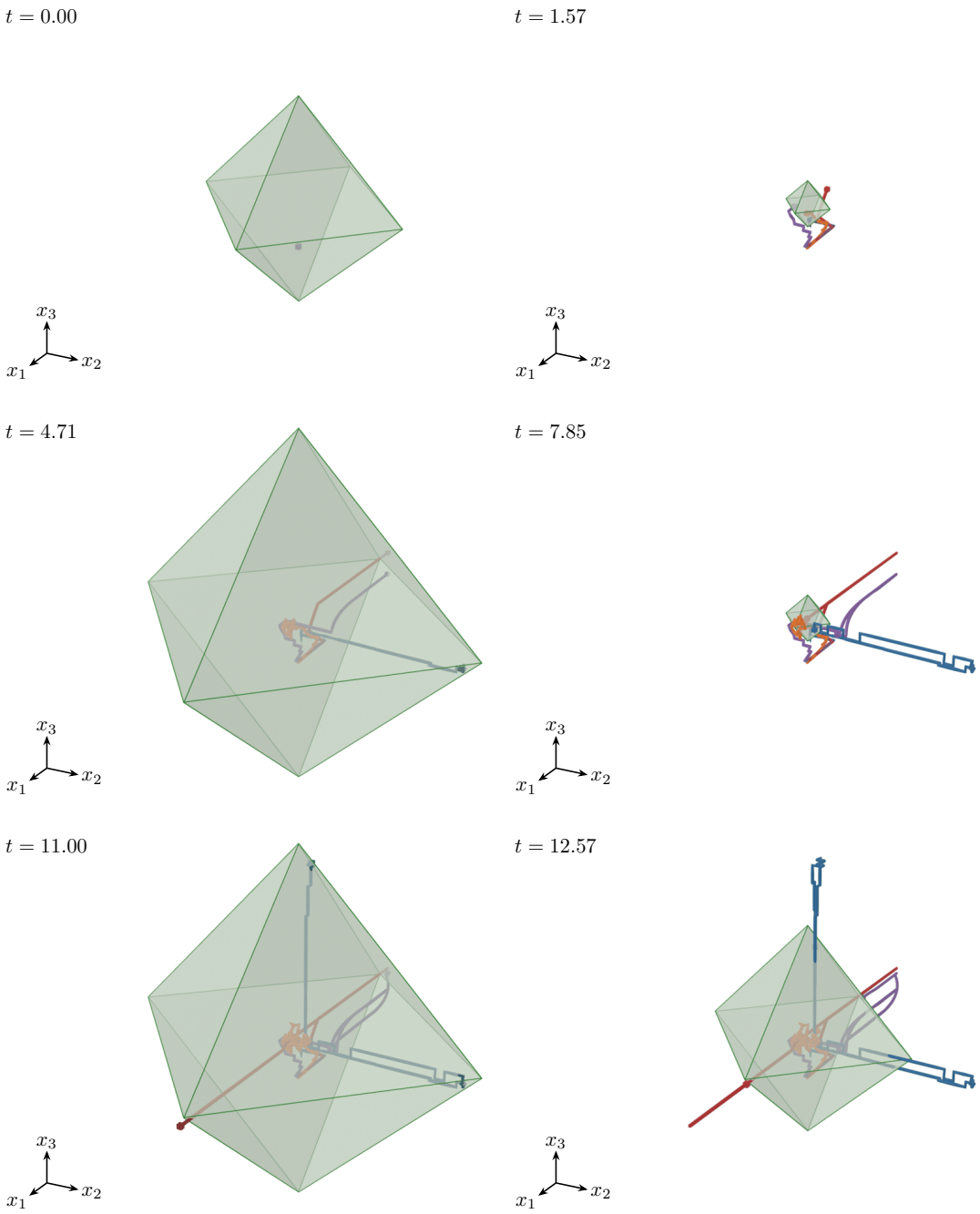


Figure 5.3: The first player's extremal strategy against the four strategies of the second.

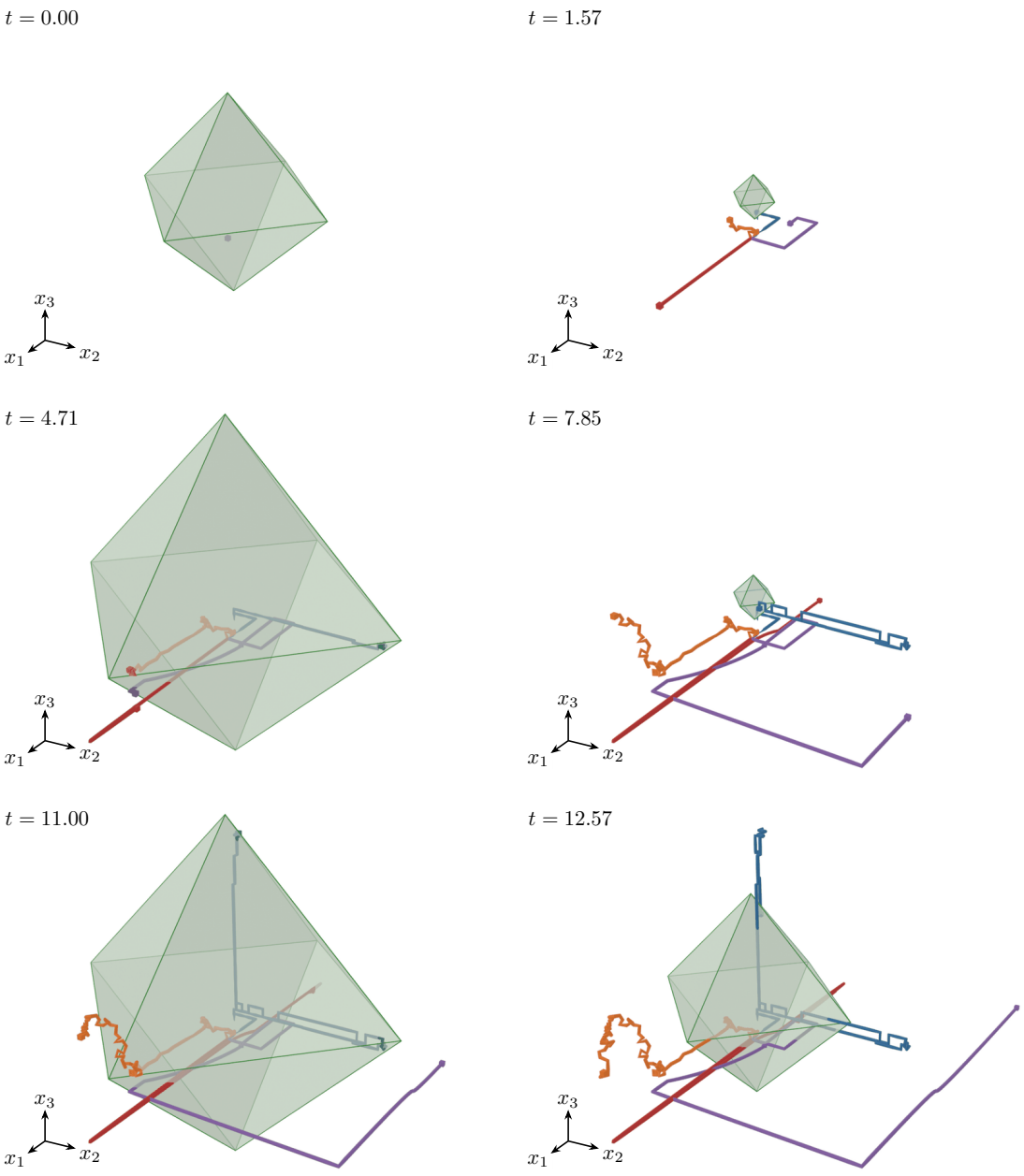


Figure 5.4: The second player's extremal strategy against the four strategies of the first.

pair of extremal strategies the terminal state is also near the boundary, within the discreteness of the scheme, as in the previous example. Under the random control, the state moves away from the sections in short steps. At the terminal instant it is outside the terminal set. Under the externally defined strategy, the state moves in wide sweeps, and the corresponding terminal state is the farthest of the four from the terminal set. These outcomes agree with the theory: the forecast state x_0 is located strictly inside the section $W(t_0)$, so the initial position is outside the bridge of the second player, and the extremal strategy gives that player no guarantee.

6 Conclusion

A simulation procedure has been described and implemented for linear differential games in which the time sections of the maximal stable bridge are polytopes given by their face lattices. The controls of each player are generated by one of four strategies, chosen independently for the two players: the extremal strategy, constant control, random control, and an externally defined strategy. On the polytopal sections, the extremal strategy locates the state by the hyperplane representation. Inside the section, the aiming point is an orthogonal projection onto a facet hyperplane. Outside it, the aiming point is found by a search over the face lattice. The cost of each step has been estimated. The random control is sampled uniformly from a face of a prescribed dimension of the constraint polytope by a descent over an auxiliary graph built from the face lattice.

The simulation has been illustrated by examples with two and three oscillators, in which the extremal strategy of each player is tested against all four strategies of the other. The outcomes agree with the guarantee of the extremal strategy to within the discretization error of the scheme, which diminishes as the time grid is refined. One direction for further work is to identify problems that can be treated with the developed library and the simulation program.

References

[1] M. Bardi, M. Falcone, and P. Soravia. Numerical methods for pursuit-evasion games via viscosity solutions. In M. Bardi, T. E. S. Raghavan, and T. Parthasarathy, editors, *Stochastic and Differential Games: Theory and Numerical Methods*, pages 105–175. Birkhäuser, Boston, 1999.

[2] N. Botkin, K.-H. Hoffmann, N. Mayer, and V. Turova. Computation of value functions in nonlinear differential games with state constraints. In D. Hömberg and F. Tröltzsch, editors, *System Modeling and Optimization: 25th IFIP TC 7 Conference, CSMO 2011, Berlin, Germany, September 12–16, 2011, Revised Selected Papers*, volume 391 of *IFIP Advances in Information and Communication Technology*, pages 235–244, Berlin, Heidelberg, 2013. Springer.

[3] N. D. Botkin and E. A. Ryazantseva. An algorithm of constructing the solvability set for linear differential games of high dimension. *Trudy Instituta Matematiki i Mekhaniki UrO RAN*, 2:128–134, 1992. (in Russian).

[4] W. Bruns and B. Ichim. Polytope volume by descent in the face lattice and applications in social choice. *Mathematical Programming Computation*, 13(2):415–442, 2021.

[5] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Some algorithms for differential games with two players and one target. *Modélisation mathématique et analyse numérique*, 28(4):441–461, 1994.

[6] S. Das, S. Ranjan Dev, and S. Sarvottamananda. A worst-case optimal algorithm to compute the minkowski sum of convex polytopes. In *Algorithms and Discrete Applied Mathematics*, volume 12601 of *Lecture Notes in Computer Science*, pages 179–195. Springer International Publishing, 2021.

[7] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986.

[8] M. V. Dolgopolik. An acceleration technique for methods for finding the nearest point in a polytope and computing the distance between two polytopes. *Communications in Optimization Theory*, 2026(4):1–37, 2026.

[9] S. A. Ganebny, S. S. Kumkov, and V. S. Patsko. Feedback control in problems with unknown level of dynamic disturbance. In F. L. Chernousko, editor, *Advances in Mechanics: Dynamics and Control. Proc. 14th International Workshop on Dynamics and Control*, pages 125–132, Moscow, 2008. Nauka.

[10] N. L. Grigorenko, Yu. N. Kiselev, N. V. Lagunova, D. B. Silin, and N. G. Trin'ko. Solution methods for two-player differential games. *Computational Mathematics and Modeling*, 8(1):34–48, 1997.

[11] R. Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. Dover Publications, Mineola, NY, 1999.

[12] E. A. Isakova, G. V. Logunova, and V. S. Patsko. Computation of stable bridges for linear differential games with fixed time of termination. In A. I. Subbotin and V. S. Patsko, editors, *Algorithms and Programs for Solving Linear Differential Games*, pages 127–158. Institute of Mathematics and Mechanics, Ural Scientific Center, Academy of Sciences of the USSR, Sverdlovsk, 1984. (in Russian).

[13] G. E. Ivanov and E. S. Polovinkin. On strongly convex linear differential games. *Differential Equations*, 31(10):1603–1612, 1995.

[14] D. V. Kornev. Numerical methods for solving differential games with nonterminal payoff. *Izvestiya Instituta Matematiki i Informatiki Udmurtskogo Gosudarstvennogo Universiteta*, (2 (48)):82–151, 2016. (in Russian).

[15] N. N. Krasovskii and A. I. Subbotin. *Positional Differential Games [Pozitsionnye differentsialnye igry]*. Nauka, Moscow, 1974. (in Russian).

[16] N. N. Krasovskii and A. I. Subbotin. *Game-Theoretical Control Problems*. Springer Series in Soviet Mathematics. Springer-Verlag, New York, 1988.

[17] A. V. Mikhailov and S. S. Kumkov. Linear differential games with multi-dimensional terminal target set: Geometric approach. In V. L. Turova, A. E. Kovtanyuk, and J. Zimmer, editors, *Proceedings of the 3rd International Workshop on Mathematical Modeling and Scientific Computing*, volume 104 of *EPiC Series in Computing*, pages 221–242, 2024.

[18] E. S. Polovinkin. Nonautonomous differential games. *Differentsial'nye Uravneniya*, 15(6):1007–1017, 1979. (in Russian).

[19] E. S. Polovinkin, G. E. Ivanov, M. V. Balashov, R. V. Konstantinov, and A. V. Khorev. An algorithm for the numerical solution of linear differential games. *Sbornik: Mathematics*, 192(10):1515–1542, 2001.

[20] L. S. Pontryagin. On linear differential games. II. *Soviet Mathematics. Doklady*, 8:910–912, 1967.

[21] L. S. Pontryagin. Linear differential games of pursuit. *Mathematics of the USSR-Sbornik*, 40(3):285–303, 1981.

[22] G. Swart. Finding the convex hull facet by facet. *Journal of Algorithms*, 6(1):17–48, 1985.

[23] M. A. Zarkh and A. G. Ivanov. Construction of the value function in the linear differential game with the fixed terminal time. *Trudy Instituta Matematiki i Mekhaniki UrO RAN*, 2:140–155, 1992. (in Russian).

[24] G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer, New York, 1995.