Non-Archimedean zero-sum games

Non-Archimedean Zero-Sum Games

Marco Cococcioni[a,*], Lorenzo Fiaschi[a], Luca Lambertini[b]

[a] *Department of Information Engineering, University of Pisa, Largo Lucio Lazzarino 1 – 56122 Pisa, Italy*
[b] *Department of Economics, University of Bologna, Strada Maggiore 45 – 40125 Bologna, Italy*

Abstract

Zero-sum games are a well known class of game theoretic models, which are widely used in several economics and engineering applications. It is known that any two-player finite zero-sum game in mixed-strategies can be solved, i.e., one of its Nash equilibria can be found solving a linear programming problem associated to it. The idea of this work is to propose and solve zero-sum games which involve infinite and infinitesimal payoffs too, that is non-Archimedean payoffs. Since to find a Nash equilibrium a non-Archimedean linear programming problem needs to be solved, we implement and extend a more powerful version of an already existing non-Archimedean Simplex algorithm, namely the Gross-Simplex one. In particular, the new algorithm, called Gross-Matrix-Simplex, is able to handle the constraint matrix A when it is made of non-Archimedean quantities. To test the correctness and the efficiency of the Gross-Matrix-Simplex algorithm, we provide four numerical experiments, which have been run on an Infinity Computer simulator. Furthermore, we also discuss several examples based on well known models related to economics, politics and engineering, where a non-Archimedean zero-sum model appears to be a reasonable, powerful and flexible representation.

Keywords: Game Theory, Zero-Sum Matrix Games, Linear Programming, Non-Archimedean Analysis, Grossone Methodology, Infinity Computer

1. Introduction

This work aims to show a contact point between three apparently separated research fields, paving the way for more interesting studies in this direction. The three topics we bring into play are three different branches of Mathematics, namely Game Theory (GT), Linear Programming (LP), and Non-Archimedean Analysis (NAA).

GT is the discipline which models the behaviour of rational agents within competitive environments, commonly called games [23]. In the current context, we narrow this topic to the case of two-player finite zero-sum games in mixed-strategies [24], that is competitive environments involving just two agents, each endowed with a finite number of possible strategies. The label "zero-sum" refers to the fact that one agent's gain corresponds to an equal loss

*Corresponding author
Email addresses: marco.cococcioni@unipi.it (Marco Cococcioni), lorenzo.fiaschi@phd.unipi.it (Lorenzo Fiaschi), luca.lambertini@unibo.it (Luca Lambertini)

December 22, 2021

for the opponent, while "mixed-strategies" means that the adoption of each strategy is modeled by a probability distribution. In order to ease reading, hereinafter we will refer to such class of games as the set $\mathcal{ZG}$.

LP is a branch of optimization theory which aims to minimize (or maximize) linear functions, subject to linear constraints (both equalities and inequalities) [35]. It is known, even if it is not straightforward, that any two-person game in $\mathcal{ZG}$ can be always transformed in an LP problem [1] whose optima are Nash equilibria of the original game. In particular, such optimization task has the game's payoffs matrix as the constraint matrix of the problem.

Finally, NAA is the research field which deals with mathematical objects lacking of the Archimedes' property [3, 28]. An algebraic structure lacking of such property contains elements that are not finitely comparable, i.e., it also contains objects that are infinitely large or infinitely small. Two examples are the Levi-Civita field [4, 22] and Robinson's hyperreal numbers [27]: both allow one to use numbers which can be infinite, infinitesimal, rather than just finite as one may be used to work with. Recently, the Grossone Methodology (GM) proposed by Y.D. Sergeyev [31] (along with his patented Infinity Computer [17, 29]) has been used to solve a number of non-Archimedean problems, or standard problems approached through a non-Archimedean perspective. Grossone Methodology is an original idea, independent from Robinson's hyperreal numbers (as discussed in [32]), and more oriented towards numerical computations.

In particular, in the last few years, the GM has been used both in LP [11, 12, 13] and in GT [24], but separately. The application of the GM to GT, in particular, has been used to model both infinite/infinitesimal payoffs and finite/infinitesimal probabilities in Prisoner's Dilemmas [18, 19, 20]. An application of GM to games on graphs appears in [10], while it has been used to numerically deal with infinite decision-making processes in [26].

On the other hand, the application of GM to LP has been addressed in [5, 6, 8, 9]. In particular, the algorithm proposed in [8], called Gross-Simplex (G-Simplex in short), is the starting point of the present work. Also the Big-M method [2, 11, 34] enjoyed the numerical advent of GM, since their encounter gave rise to the so-called Infinitely-Big-M method (I-Big-M) [7].

All in all, the motivation of this work is to solve games in $\mathcal{NZG}$, i.e., the non-Archimedean extension of $\mathcal{ZG}$ by means of a new algorithm, called Gross-Matrix-Simplex (in brief GM-S).

The latter is able to solve non-Archimedean LP problems also having the constraint matrix and the unknowns vector which are filled with non-Archimedean numbers. In order to show the effectiveness of the proposed approach, we run several experiments whose results are reported in this letter. Such tests are also aimed to stress the numerical computability aspect of our work, even if the problems to solve and the algorithm to run involve infinite and infinitesimal numbers. In particular, all the routines have been implemented in software and launched on an Infinity Computer simulator implemented in the same language.

The remainder of the paper is structured as follows: Section 2 introduces the zero-sum games in $\mathcal{ZG}$, Section 3 contains a brief discussion about GM, while Section 4 shows how GM-S algorithm is a feasible choice to solve non-Archimedean zero-sum games in $\mathcal{NZG}$. Then, Section 5 presents numerical experiments verifying the effictivenesss of the method, and Section 6 paves the way for future work, suggesting possible applications in several fields, where

the ideas illustrated in this work could be fruitfully applied. Concluding remarks are in Section 7.

2. Zero-Sum Games

A non-cooperative game $\mathcal{G}$ is a mathematical model designed to formally describe the selfish interaction of a set of rational agents within an environment. Commonly, $\mathcal{G}$ is represented by means of the triple $\{u, S, N\}$, where $N = \{1, \ldots, n\}$ indicates the set of agents which take part to the game. Each agent i, also known as player, has at his/her disposal the set S_i of strategies to adopt within the game. Then, the set S is defined as the Cartesian product of all the players sets of strategies, i.e., $S = \prod_{i=1}^{n} S_i$. Moreover, each agent i brings along his/her own utility function $u_i : S \to \mathbb{R}$, which describes the player income provided the strategies implemented by all the participants. Therefore, $u : S \to \mathbb{R}^n$ is the function which has u_i as the i-th component.

A game $\mathcal{G}$ can be classified in accordance to several orthogonal rules. For instance, $\mathcal{G}$ is addressed as a zero-sum game [36] whenever it satisfies the following property:

$$\sum_{i=1}^{n} u_i(s) = 0 \ \ \forall\, s \in S.$$

The latter means that the total utility drained out of the game by all the players is always zero (whence the zero-sum label). On the other hand, a game is said to be *finite* if it describes a competitive scenario where each agent can choose among a finite number of admissible strategies, i.e.,

$$|S_i| \in \mathbb{N} \ \ \forall\, i \in N.$$

Such class of games falls under the name of *matrix games*, since they can be fully represented by an n-dimensional matrix A such that $A = [u(s_k)]_k$, where $k = (k_1, \ldots, k_n)$, $s_k = (s_{k_1}, \ldots, s_{k_n})$, $k_i \in \{1, \ldots, |S_i|\}$ and $s_{k_i} \in S_i$, $i = 1, \ldots, n$. Whenever the game involves just two players, i.e., $n = 2$ the matrix A (also known as matrix of the game) simplifies a lot, and it can be written as follows

$$A = \begin{bmatrix} u(s_1^1,\, s_1^2) & \cdots & u(s_1^1, s_{|S_2|}^2) \\ \vdots & \ddots & \vdots \\ u(s_{|S_1|}^1,\, s_1^2) & \cdots & u(s_{|S_1|}^1, s_{|S_2|}^2) \end{bmatrix},$$

where s_j^i means the j-th strategy of player i, i.e., $s_j \in S_i$, $i = 1, 2$. Finally, a game $G' = \{h, \Delta, N\}$ is said to be played in *mixed-strategies* when there exists a game $G = \{u, S, N\}$ and the following equalities are satisfied:

$$\Delta_{S_i} \coloneqq \left\{ \delta_i : S_i \to [0,1] \ : \ \sum_{s \in S_i} \delta_i(s) = 1 \right\}, \quad \Delta \coloneqq \prod_{i=1}^{n} \Delta_{S_i}$$

$$\delta : S \to [0,1], \quad \delta(s) = \prod_{i=1}^{n} \delta_i(s_i), \quad \delta_i \in \Delta_{S_i}, \quad s_i \in S_i$$

$$h_i(\delta) \coloneqq \sum_{s \in S} u_i(s)\delta(s), \quad h \coloneqq (h_1, \ldots, h_n).$$

The above notation means that the agents in $\mathcal{G}'$ compete in the very same environment of $\mathcal{G}$, but this time each of them can adopt a behavior δ_i which is actually a probability distribution over the strategies, rather than choosing a single one (whence the *mixed* nature of their strategies).

In this work, we will focus on two-player finite zero-sum games in mixed-strategies, which form a very peculiar class of games for three reasons. Finiteness and mixed-strategies guarantee the existence of at least one Nash equilibrium, as a corollary of the Nikaido-Isoda theorem (1955). The presence of two players and the zero-sum property jointly imply that every Nash equilibrium is at the intersection of minimax (or maximin) strategies. Together, these four properties and their consequences allow one to reformulate the search for Nash equilibria as an LP problem [1]. Indeed, the Nash equilibria of such games are all and only the mixed-strategies x and y which satisfy

$$\min_{x \in \Delta_{S_1}} \max_{y \in \Delta_{S_2}} x^T A y = \max_{x \in \Delta_{S_1}} \min_{y \in \Delta_{S_2}} x^T A y \,, \tag{1}$$

where A is the game matrix. However, the problem in (1) can be straightforwardly reformulated in an LP problem whose primal and dual are in (2) and (3), respectively

$$\begin{aligned} &\max \lambda \\ &\text{s.t. } \lambda \mathbf{1} \leq x^T A \\ &\quad x^T \mathbf{1} = 1 \\ &\quad x \geq 0 \end{aligned} \tag{2}$$

$$\begin{aligned} &\min \mu \\ &\text{s.t. } Ay \leq \mu \mathbf{1} \\ &\quad \mathbf{1}^T y = 1 \\ &\quad y \geq 0 \end{aligned} \tag{3}$$

where $\mathbf{1}$ is the vector, of proper dimension, filled by ones, and λ, $\mu \in \mathbb{R}$.

Both problems can be transformed into the following form, which is in the canonical one and, as such, can be solved through the revised simplex algorithm:

$$\begin{aligned} \min \;& c^T x \\ \text{s.t. } & Ax = b \\ & x^T \mathbf{1} = 1 \\ & x \geq 0 \end{aligned}$$

3. Grossone Methodology

GM is a novel non-Archimedean framework to deal with infinite and infinitesimal quantities in a numerical way. It has been proposed by Sergeyev, and [31] contains an exhaustive discussion of the topic. GM has found several applications in optimization theory, such as regularization [14], conjugate gradient methods [15], and especially in lexicographic multi-objective LP. In [8], indeed, a Grossone-version of the Simplex algorithm (the G-Simplex) has been implemented and theoretically studied. Such algorithm has been successfully applied in [7] to implement a non-Archimedean and parameter-less version of the Big-M method, namely I-Big-M. The latter allows one to solve

any LP problem (even lexicographic multi-objective) by means of a single run of the G-Simplex algorithm, without caring about possible initialization issues and in a manner totally transparent to the user. Also GT enjoyed the advent of GM, as can be seen in [10, 18, 20]. In particular, it has been used to model three different aspects of a game: i) players' payoffs when made up by multiple goods lexicographically ordered; ii) probability of very rare but not impossible events; iii) infinitely long intervals of time.

Within GM, the numeral ① plays a crucial role. It represents the infinite unit (for its definition see the Infinite Unit axiom in [31]), and it is used as the basis upon which to build a novel numeral system containing numbers of the following form

$$\tilde{z} = z_m ①^{p_m} + \ldots + z_0 + \ldots + z_{-k} ①^{p_{-k}}.$$

Such numbers are called *gross*-scalars (G-scalars in brief), $m, k \in \mathbb{N}$, exponents p_i are called *gross*-powers (G-powers, they are G-scalars as well), and $z_i \neq 0$ are called *gross*-digits, finite real numbers, $i = m, \ldots, -k$. The elements of such new numeral system continue to satisfy all the algebraic properties of the real numbers, that is commutative, distributive, existence of the inverse, and so on. For the sake of clarity, some examples of G-scalars algebraic manipulation follow:

$$\left(2 + ①^{-0.5}\right) ① = 2 \cdot ① + ①^{-0.5} \cdot ① = 2① + ①^{0.5},$$

$$\frac{1}{①} = ①^{-1} > 0, \quad \frac{-6①^3 + 26 - 24①^{-3}}{-2①^3 + 6} = 3 - 4①^{-3}.$$

Later on, the concept of *gross*-vector (G-vector) will be useful; it consists of a vector whose entries are G-scalars.

In this work we will leverage on G-scalars $\tilde{z}$ of the form

$$\tilde{z} = \sum_{i \in \mathbb{P}} z_i ①^i,$$

where $\mathbb{P} \subset \mathbb{Z}$ is a finite set of G-powers, $z_i \in \mathbb{R}\ \forall\, i \in \mathbb{P}$. Here, a G-scalar is infinite when $\exists\, i > 0$ s.t. $z_i \neq 0$ (i.e., at least one of its G-powers is positive), finite when it is not infinite and $z_0 \neq 0$, infinitesimal when it is not infinite nor finite. Such numbers will be useful to represent non-Archimedean payoffs of a game. For instance, consider the payoff $2① + 3 - 4①^{-1}$. It can be seen as a prize made up by three different goods lexicographically ordered, as originally suggested in [30]. In particular, it consists of two units of the most important good, followed by three of the second one and minus four of the third one. As can be seen, the priority is represented by a weighted sum where the weights span different order of infinite. Indeed, the contribution to the payoff of the most important good is scaled up by an infinite factor, i.e., ①. The one of the second good is weighted by 1 ($= ①^0$), while the third one is scaled down by $①^{-1}$, making its contribution infinitesimal. All in all, the player who receives that payoff obtains two units of the first good and three of the second one, provided he/she pays back four units of the third good.

4. Non-Archimedean Zero-Sum Games and the Gross-Matrix-Simplex Algorithm

Section 2 highlighted how games in $\mathcal{ZG}$ can be solved, i.e., one of its Nash equilibria can be found by solving an LP problem. When the zero-sum game involves non-Archimedean quantities, the problems in Equations (2) and (3) happen to be non-Archimedean as well. Thus, to solve them one needs to resort on a non-Archimedean optimizer. As stated in Section 3, a non-Archimedean Grossone-based version of a very famous algorithm to solve LP problems, namely the Simplex algorithm, already exists and is called G-Simplex algorithm. However, its actual implementation leads to solve non-Archimedean LP problems (in brief, NALP) which involve non-Archimedean cost functions only. This limitation impedes one to adopt such algorithm in the present context. Indeed, the NALP problem we are facing features non-Archimedean values in the constraints matrix, since the latter represents the payoffs matrix of a game in $\mathcal{NZG}$.

In order to tackle this wider set of NALP problems, we implement a new algorithm, called Gross-Matrix-Simplex, which can find optimal solutions that are themselves non-Archimedean. The problem to solve is the following:

$$\begin{aligned} \min \ & c^T \tilde{x} \\ \text{s.t.} \ & \tilde{A}\tilde{x} = b \\ & \tilde{x} \geq 0 \end{aligned} \tag{4}$$

where $\tilde{A}$ is the G-matrix containing the payoffs (the Gross-Matrix matrix), c and b are the (purely finite) objective and constant-terms vectors.

The algorithm to solve it is given in Algorithm 1. As anticipated, it has been named Gross-Matrix-Simplex to distinguish it from the Gross-Simplex provided in [8]. The latter, in retrospect, should have been named Gross-Cost-Simplex, since it is only able to handle non-Archimedean cost functions $\tilde{c}$ (i.e., it is able to solve only lexicographic multi-objective problems given purely finite constraints). The key ingredient in Algorithm 1 is the computation of the inverse of a non-Archimedean matrix, which has been computed using the classical Gaussian-Jordan elimination algorithm, but in this case non-Archimedean quantities are involved (in particular, the product and the division between two G-scalars). The algorithms for computing the product and the division are provided in [31]). In future, we will consider avoiding the explicit computation of the inverse, by using the incremental LU factorization, as in any optimized implementation of the Simplex algorithm.

The effectiveness of GM-S to solve NALP problems, as well as the possibility to numerically solve games in $\mathcal{NZG}$ by means of GM, have been computationally validated by means of an Infinity Computer simulator. The outputs have been collected and they are reported in the next section.

5. Numerical Illustrations

In this section, we provide some numerical experiments which illustrate the effectiveness of GM-S algorithm to solve games in $\mathcal{NZG}$. Of course, the results we find are approximated, both in terms of G-digits and number

Algorithm 1 The Gross-Matrix-Simplex algorithm

Step 0. The user has to provide the initial set $\mathtt{B}$ of basic indices.

Step 1. Compute $\tilde{x}_{\mathtt{B}}$ as: $\tilde{x}_{\mathtt{B}} = \tilde{A}_{\mathtt{B}}^{-1} b$ (where $\tilde{A}_{\mathtt{B}}$ is the sub-matrix obtained by $\tilde{A}$ by considering the columns indexed by $\mathtt{B}$, while $\tilde{A}_{\mathtt{B}}^{-1}$ is the inverse of $\tilde{A}_{\mathtt{B}}$, i.e., the non-Archimedean matrix which satisfies the equality $\tilde{A}_{\mathtt{B}}^{-1}\tilde{A}_{\mathtt{B}} = \tilde{A}_{\mathtt{B}}\tilde{A}_{\mathtt{B}}^{-1} = I$).

Step 2. Compute $\tilde{y}$ as: $\tilde{y} = c_{\mathtt{B}}^T \tilde{A}_{\mathtt{B}}^{-1}$ (where $\tilde{y}$ is a G-vector obtained by linearly combining the purely finite vector $c_{\mathtt{B}}^T$ by the G-scalar elements in the rows of $\tilde{A}_{\mathtt{B}}^{-1}$).

Step 3. Compute $\tilde{s}$ as: $\tilde{s} = c_{\mathtt{N}}^T - \tilde{y}\tilde{A}_{\mathtt{N}}$ (where $\mathtt{N}$ is the complementary set of $\mathtt{B}$) and then select the maximum (gradient rule). When this maximum is negative (being it infinite, finite or infinitesimal), then the current solution is optimal and the algorithm stops. Otherwise, the position of the maximum in the G-vector $\tilde{s}$ is the index k of the entering variable $\mathtt{N}(k)$.

Step 4. Compute $\tilde{d}$ as: $\tilde{d} = \tilde{A}_{\mathtt{B}}^{-1}\tilde{A}_{\mathtt{N}(k)}$, where $\tilde{A}_{\mathtt{N}(k)}$ is the G-vector corresponding to the $\mathtt{N}(k)$-th column of $\tilde{A}$.

Step 5. Find the largest G-scalar $\tilde{t} > 0$ such that $\tilde{x}_{\mathtt{B}} - \tilde{t}\tilde{d} \geqslant 0$. If there is not such a $\tilde{t}$, then the problem is unbounded (STOP); otherwise, at least one component of $\tilde{x}_{\mathtt{B}} - \tilde{t}\tilde{d}$, say h, equals to zero and the corresponding variable is the leaving variable.

Step 6. Update sets $\mathtt{B}$ and $\mathtt{N}$ by swapping $\mathtt{B}(h)$ and $\mathtt{N}(k)$, then return to Step 1.

of components representing them. Indeed, as in any numerical algorithm, when the finite precision used to do computation is not enough to contain all the information, only the highest informative components of data are held, while the remaining ones are discarded. Actually, it is perfectly reasonable that some non-Archimedean Nash equilibria may need an infinite number of components to be exactly represented, as well as in the standard domain they may need an infinite number of digits.

Most of the experiments below refer to non-Archimedean variations of the famous rock-paper-scissors game, a very well known example of zero-sum game. We decided to focus on this model since it is simple enough to make its non-Archimedean modifications easy to understand and predictable in terms of Nash equilibria alterations. Its canonical form is reported in Table 1:

Table 1: Rock-Paper-Scissors canonical form

Algorithm 2 Procedure to double-check the results provided by the GM-S algorithm

step 0. Let (x^*, y^*) be a Nash equilibrium for a given n-dimensional game obtained somehow (in our case by GM-S).

Step 1. Indicate with $\mathcal{A}_x$ and $\mathcal{A}_y$ the index set of active strategies in x^* and y^*, respectively. This means that $i \in \mathcal{A}_z \Leftrightarrow z_i > 0$, $i \in \{1, \ldots, n\}$, $z = x, y$

Step 2. Define the active matrix B as the payoff matrix reduced to the row indexes in $\mathcal{A}_x$ and the column indexes in $\mathcal{A}_y$, i.e., $B \coloneqq A^{\mathcal{A}_y}_{\mathcal{A}_x}$.

Step 3. Define C and D such that $C_i = B^i - B^{i+1}\ \forall\, i = 1, \ldots, |\mathcal{A}_y| - 1$ and $C_{|\mathcal{A}_y|} = \mathbf{1}$, while for D holds true that $D_i = B_i - B_{i+1}\ \forall\, i = 1, \ldots, |\mathcal{A}_x| - 1$ and $D_{|\mathcal{A}_x|} = \mathbf{1}$.

Step 4. Verify that $Cx^* = e$ and $Dy^* = e$ hold true, where we have indicated with e the last (under the natural ordering) vector of the canonical base with proper dimension, i.e., $e = (0, \ldots, 0, 1)^T$.

$\wp_1$ \ $\wp_2$	**R**	**P**	**S**
R	0	1	-1
P	-1	0	1
S	1	-1	0

The mathematical formulation of the problem is reported in (5), where A indicates the payoffs matrix (the content of A coincides with the content of Table 1):

$$\min_{x \in \Delta_1} \max_{y \in \Delta_2} x^T A y \tag{5}$$

$$\Delta_1 = \Delta_2 = \left\{ (\rho_1, \rho_2, \rho_3) \in \mathbb{R}^3 \;\middle|\; \sum_{i=1}^{3} \rho_i = 1,\ \rho_i \geq 0\, \forall\, i \right\}$$

As widely known, in this game there exists only one Nash equilibrium, which is shown in Equation (6):

$$x^* = y^* = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3} \right] \tag{6}$$

For the sake of consistency, we mention that all the experiments outcomes have been double-checked verifying that they are *basic* Nash equilibria [33]. The procedure to follow in order to check it is reported in Algorithm 2.

5.1. Experiment 1: Infinitesimally perturbed rock-paper-scissors

The first experiment to validate our efforts is to adopt GM-S to solve the very same problem of Table 1, but considering an infinitesimal perturbation on one of the payoffs. Such experiment is meant to show the sensibility

of GM-S algorithm to infinitesimal changes in the matrix game. As it can be seen from Equation (7), the entry $\tilde{A}_{2,1}$ has been infinitesimally altered (to improve the readability, from now on the infinitesimal components will be colored in blue). We can expect the new equilibrium differs from the one in (6), but our intuition tells us that it should be a point $(\tilde{x}^*, \tilde{y}^*)$ infinitely close to it. Indeed, from a finite perspective the game is unchanged, thus, it is reasonable to expect that the Nash equilibrium's finite digits will be unchanged as well (this is true since the Nash equilibrium is unique). In Table 2, we reported the iterations executed during the optimization, while in Equation (8) the new Nash equilibrium found by GM-S algorithm. As expected, the latter is infinitely close to the one in Equation (6), but GM-S tells us *exactly how much*.

$$\tilde{A} = \begin{bmatrix} 0 & 1 & -1 \\ -1-①^{-1} & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \tag{7}$$

Table 2: GM-S iterations for Game 1 of Equation (7): infinitesimally perturbed rock-paper-scissors game

It.	**Base**	$\tilde{x}$	$c^T\tilde{x}$
1	$\{4, 5, 6\}$	$\left[0, 0, 0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0\right]$	$-①$
2	$\{2, 5, 6\}$	$\left[0, \frac{1}{4} - \frac{1}{8}①^{-1} + \frac{1}{16}①^{-2}, 0, 0, \frac{1}{4} + \frac{1}{8}①^{-1} - \frac{1}{16}①^{-2}, \frac{1}{2}, 0, 0, 0\right]$	$-\frac{3}{4}① - \frac{3}{8} + \frac{3}{16}①^{-1} - \frac{1}{16}①^{-2}$
3	$\{2, 3, 6\}$	$\left[0, \frac{1}{3} - \frac{1}{9}①^{-1} + \frac{1}{27}①^{-2}, \frac{1}{6} + \frac{1}{9}①^{-1} - \frac{1}{27}①^{-2}, 0, 0, \frac{1}{2}, 0, 0, 0\right]$	$-\frac{1}{2}① - \frac{1}{2}$
4	$\{2, 3, 1\}$	$\left[\frac{1}{3}, \frac{1}{3} - \frac{1}{9}①^{-1} + \frac{1}{27}①^{-2}, \frac{1}{3} + \frac{1}{9}①^{-1} - \frac{1}{27}①^{-2}, 0, 0, 0, 0, 0, 0\right]$	-1

$$\tilde{x}^* = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} - \frac{1}{9}①^{-1} + \frac{1}{27}①^{-2} \\ \frac{1}{3} + \frac{1}{9}①^{-1} - \frac{1}{27}①^{-2} \end{bmatrix}, \qquad \tilde{y}^* = \begin{bmatrix} \frac{1}{3} - \frac{1}{9}①^{-1} + \frac{1}{27}①^{-2} \\ \frac{1}{3} \\ \frac{1}{3} + \frac{1}{9}①^{-1} - \frac{1}{27}①^{-2} \end{bmatrix} \tag{8}$$

From Table 2 it can also be seen that the cost function starts from an infinite value and that the length of vector $\tilde{x}$ is larger than expected. This is due to the fact that here we use the same strategy used in the design of the I-Big-M method [7], i.e., we have added to the problem a set of artificial variables, which have been infinitely penalized (by the term ①, as done in the pioneering works [9, 14] by De Leone et al.). This allowed us to have an initial basis to start from (the one made of only artificial variables). At the end of the optimization, the cost function has a finite value, meaning that all the artificial variables have exited the base. Notice that this also explains the larger number of components of vector $\tilde{x}$: the additional entries are the values of the artificial variables.

5.2. Experiment 2: A purely finite 4-by-3 game

Now, we move on to consider an experiment where even an infinitesimal perturbation can significantly affect the choice of which Nash equilibrium to play, a rather counterintuitive phenomenon. We crafted it starting from the rock-paper-scissors game of Table 5 and adding to it an extra strategy for the row player. The resulting game

is shown in Equation (9), and it appears clearly as a standard zero-sum game in $\mathcal{ZG}$. The crucial property of such game is that it is characterized by infinitely many Nash equilibria, without any further criterion by means of which to choose among them. Thus, the choice of which mixed-strategy to play just depends on the arbitrary choice induced by the actual implementation of the decision algorithm adopted (in our case, the Simplex algorithm along with its specific implementation).

$$A = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \\ \frac{1}{2} & -1 & \frac{1}{2} \end{bmatrix} \tag{9}$$

In Table 3, we reported the iterations of GM-S algorithm run on the degenerate problem above (which is made of purely finite numbers). As it can be seen, the routine returns the new row-player optimal strategy $x^* = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0]^T$, which, however, is essentially the same one in the original rock-paper-scissors given in Equation (6), i.e., $x^* = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T$. Indeed, the added strategy plays no role in this case. Next experiment concerns an infinitesimal perturbation on the payoffs used in the current experiment. We will show how such an infinitesimal perturbation does not cause an infinitesimal change on the equilibrium, as one might expect, but a finite one.

Table 3: GM-S iterations for the purely finite Game 2 of Equation (9)

It.	**Base**	$\tilde{x}$	$c^T\tilde{x}$
1	$\{5,\ 6,\ 7\}$	$\left[0,\ 0,\ 0,\ 0,\ \frac{1}{3},\ \frac{1}{3},\ \frac{1}{3},\ 0,\ 0,\ 0\right]$	$-①$
2	$\{5,\ 6,\ 1\}$	$\left[\frac{1}{4},\ 0,\ 0,\ 0,\ \frac{1}{4},\ \frac{1}{2},\ 0,\ 0,\ 0,\ 0\right]$	$-\frac{3}{4}① - \frac{1}{4}$
3	$\{2,\ 6,\ 1\}$	$\left[\frac{1}{3},\ \frac{1}{6},\ 0,\ 0,\ 0,\ \frac{1}{2},\ 0,\ 0,\ 0,\ 0\right]$	$-\frac{1}{2}① - \frac{1}{2}$
4	$\{2,\ 3,\ 1\}$	$\left[\frac{1}{3},\ \frac{1}{3},\ \frac{1}{3},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right]$	-1

5.3. Experiment 3: Infinitesimally perturbed 4-by-3 game

As anticipated, we now assume that we want to embed a secondary information in the game. Such additional information consists of a second payoff matrix A_s whose importance is infinitely less than the one of Equation (9):

$$A_s = \begin{bmatrix} 0 & 2 & -2 \\ -2 & 0 & 2 \\ 2 & -2 & 0 \\ -1 & -2 & 1 \end{bmatrix}$$

A possible way to combine the two pieces of information, the principal and the secondary one, within a single zero-sum game may be to sum the two matrices into a new non-Archimedean one. In particular, the new payoff matrix is built filling each of its entries with a G-number whose finite component is the corresponding entry of

the primary payoff matrix, and whose infinitesimal component is the corresponding entry of the secondary payoff matrix, i.e.:

$$\tilde{A} = A + A_s①^{-1}$$

Thus the new payoff matrix becomes

$$\tilde{A} = \begin{bmatrix} 0 & 1 + 2①^{-1} & -1 - 2①^{-1} \\ -1 - 2①^{-1} & 0 & 1 + 2①^{-1} \\ 1 + 2①^{-1} & -1 - 2①^{-1} & 0 \\ \frac{1}{2} - ①^{-1} & -1 - 2①^{-1} & \frac{1}{2} + ①^{-1} \end{bmatrix} \tag{10}$$

The overall game is now a non-Archimedean one, falling in the set $\mathcal{NZG}$, since $\tilde{A}$ is non-Archimedean.

Even more importantly, we will see that this time the perturbation has a very notable impact on the whole game, despite the infinitesimal perturbation on the matrix of the payoffs. Indeed, we are no longer in a context similar to that of Table 1. There, the Nash equilibrium was unique, implying that an infinitesimal perturbation on it was only able to infinitesimally alter it. Here, there exist infinitely many Nash equilibria perfectly identical from a finite optimization perspective. Thus, the presence of a secondary information puts the decision maker, i.e., the optimization algorithm in a position to single out the preferable ones also under the additional information (the infinitesimal payoff matrix A_s). This implies that an algorithm which is able to deal with a secondary information can find a better Nash equilibrium than the one in the standard case. More importantly, the former can be notably far from the latter. Indeed, they may differ by finite quantities in the face of an infinitesimal perturbation. To confirm this assertion, we report GM-S algorithm iterations computed to solve the problem with secondary information as in Equation (10). In particular, Table 4 shows the algorithm's steps for the row-player, while Table 5 the steps for the column-one.

The new equilibrium is:

Table 4: GM-S iterations for Game 3 of Equation (10): row-player

It.	**Base**	$\tilde{x}$	$c^T\tilde{x}$
1	{5, 6, 7}	$\left[0, 0, 0, 0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0\right]$	$-①$
2	{5, 4, 7}	$\left[0, 0, 0, \frac{1}{4} - \frac{1}{4}①^{-1} + \frac{1}{4}①^{-2}, \frac{3}{8} - \frac{1}{8}①^{-1} + \frac{1}{8}①^{-2}, 0, \frac{3}{8} + \frac{3}{8}①^{-1} - \frac{3}{8}①^{-2}, 0, 0, 0\right]$	$-\frac{3}{4}① - \frac{1}{2} + \frac{1}{2}①^{-1} - \frac{1}{4}①^{-2}$
3	{5, 4, 1}	$\left[\frac{3}{10} + \frac{3}{50}①^{-1} + \frac{3}{250}①^{-2}, 0, 0, \frac{2}{5} + \frac{2}{25}①^{-1} + \frac{2}{125}①^{-2}, \frac{3}{10} - \frac{7}{50}①^{-1} - \frac{7}{250}①^{-2}, 0, 0, 0, 0, 0\right]$	$-\frac{3}{10}① - \frac{14}{25} - \frac{14}{125}①^{-1} - \frac{3}{250}①^{-2}$
4	{2, 4, 1}	$\left[\frac{2}{5} + \frac{4}{75}①^{-1} - \frac{16}{25}①^{-2}, \frac{1}{5} - \frac{28}{75}①^{-1} + \frac{56}{125}①^{-2}, 0, \frac{2}{5} + \frac{8}{25}①^{-1} - \frac{48}{125}①^{-2}, 0, 0, 0, 0, 0, 0\right]$	-1

$$\tilde{x}^* = \begin{bmatrix} \frac{2}{5} + \frac{4}{75}①^{-1} - \frac{16}{25}①^{-2} \\ \frac{1}{5} - \frac{28}{75}①^{-1} + \frac{56}{125}①^{-2} \\ 0 \\ \frac{2}{5} + \frac{8}{25}①^{-1} - \frac{28}{125}①^{-2} \end{bmatrix}, \qquad \tilde{y}^* = \begin{bmatrix} \frac{1}{3} + \frac{4}{15}①^{-1} - \frac{8}{25}①^{-2} \\ \frac{1}{3} - \frac{4}{15}①^{-1} + \frac{8}{25}①^{-2} \\ \frac{1}{3} \end{bmatrix} \tag{11}$$

Table 5: GM-S iterations for Game 3 of Equation (10): column-player

It.	Base	$\tilde{y}$	$c^T\tilde{y}$
1	{4, 5, 6, 7}	$\left[0,\ 0,\ 0,\ \frac{1}{4},\ \frac{1}{4},\ \frac{1}{4},\ \frac{1}{4},\ 0,\ 0,\ 0,\ 0\right]$	$-①$
2	{4, 3, 6, 7}	$\left[0,\ 0,\ \frac{2}{9}-\frac{17}{5}①^{-1}+\frac{27}{50}①^{-2},\ \frac{4}{9}+\frac{1}{5}①^{-1}-\frac{31}{100}①^{-2},\ 0,\ \frac{2}{9}+\frac{1}{10}①^{-1}-\frac{3}{20}①^{-2},\ \frac{1}{9}+\frac{1}{25}①^{-1}-\frac{2}{25}①^{-2},\ 0,\ 0,\ 0,\ 0\right]$	$-\frac{7}{9}①+\frac{253}{450}+\frac{394}{100}①^{-1}-\frac{27}{50}①^{-2}$
3	{4, 3, 6, 1}	$\left[\frac{1}{10}+\frac{1}{25}①^{-1}-\frac{8}{125}①^{-2},\ 0,\ \frac{3}{10}-\frac{7}{25}①^{-1}+\frac{56}{125}①^{-2},\ \frac{1}{2}+\frac{2}{5}①^{-1}-\frac{6}{25}①^{-2},\ 0,\ \frac{1}{10}-\frac{4}{25}①^{-1}-\frac{18}{125}①^{-2},\ 0,\ 0,\ 0,\ 0,\ 0\right]$	$-\frac{3}{5}①-\frac{16}{25}+\frac{78}{125}①^{-1}-\frac{48}{125}①^{-2}$
4	{4, 3, 2, 1}	$\left[\frac{1}{3}-\frac{4}{3}①^{-1}+\frac{16}{3}①^{-2},\ \frac{1}{3}-\frac{8}{3}①^{-1}+\frac{40}{3}①^{-2},\ \frac{1}{3}-\frac{8}{3}①^{-2},\ 4①^{-1}-16①^{-2},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right]$	$-5+20①^{-1}-16①^{-2}$
5	{7, 3, 2, 1}	$\left[\frac{1}{3}-\frac{2}{9}①^{-1}+\frac{4}{27}①^{-2},\ \frac{1}{3}-\frac{2}{9}①^{-1}+\frac{4}{27}①^{-2},\ \frac{1}{3}-\frac{2}{9}①^{-1}+\frac{4}{27}①^{-2},\ 0,\ 0,\ 0,\ \frac{2}{3}①^{-1}-\frac{4}{9}①^{-2},\ 0,\ 0,\ 0,\ 0\right]$	$-\frac{5}{3}+\frac{10}{9}①^{-1}-\frac{4}{9}①^{-2}$
6	{10, 3, 2, 1}	$\left[\frac{1}{3}-\frac{8}{15}①^{-2},\ \frac{1}{3}-\frac{8}{15}①^{-1}+\frac{8}{15}①^{-2},\ \frac{1}{3}-\frac{8}{39}①^{-1},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ \frac{4}{5}①^{-1},\ 0\right]$	$-1+\frac{4}{5}①^{-1}$

Observe how the finite part of it has changed from $x^* = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0]^T$ to $x^* = [\frac{2}{5}, \frac{1}{5}, 0, \frac{2}{5}]^T$.

5.4. Experiment 4: high dimensional game

To demonstrate the ability of our method to solve much more complex games (thanks to its numerical, not symbolic nature), we have run two high-dimensional experiments. The first one involves just 8 strategies for each player, thus it is not a true high-dimensional problem. However, we decided to introduce it the same in order to stress how symbolic tools such as Mathematica start to struggle when game complexity grows, even with with not so high dimensional problems such as non-Archimedean ones involving an 8×8 payoff matrix and two infinitesimal components for each entry. Indeed, a symbolic tool could be used to solve the experiments above, even if it would need the repeated use of Algorithm 2 with randomly chosen active strategies, as stated in [33], which is an unpractical, combinatorial, NP-hard task. Even if found, the Nash equilibrium readability would probably be quite low, as can be seen in Figure 1 where is reported the first entry of the row player optimal strategy computed in Mathematica (the letter g stands for ①). On the other hand, the numerical and approximated tool GM-S is able to show the equilibrium components in a very interpretable way, see for instance Equation (12) where the solution of the 8×8 problem is shown. The payoff matrix and GM-S iterations can be found in Appendix A.

$$\tilde{x}^* = \begin{bmatrix} 0.18\ -0.36①^{-1} + 1.05①^{-2} \\ 0.15\ +0.01①^{-1} - 1.96①^{-2} \\ 0.26\ -0.66①^{-1} + 1.34①^{-2} \\ 0 \\ 0.22\ +0.3①^{-1} + 2.09①^{-2} \\ 0.08\ +0.36①^{-1} - 0.95①^{-2} \\ 0.11\ +0.34①^{-1} - 1.57①^{-2} \\ 0 \end{bmatrix}, \qquad \tilde{y}^* = \begin{bmatrix} 0.01\ +0.54①^{-1} + 3.12①^{-2} \\ 0.14\ -0.15①^{-1} - 1.03①^{-2} \\ 0 \\ 0.21\ +0.37①^{-1} + 2.71①^{-2} \\ 0.27\ +0.34①^{-1} - 0.81①^{-2} \\ 0 \\ 0.1\ -0.81①^{-1} - 2.19①^{-2} \\ 0.27\ -0.29①^{-1} - 1.8①^{-2} \end{bmatrix} \tag{12}$$

When the payoff matrix is 10×10, instead, the symbolic tool struggles in computing the game solution and the execution time grows drastically, passing from 1.28 seconds of the 8×8 game to 6.98. To stress even more this fact, we also considered a true high-dimensional game, in fact a non-Archimedean zero-sum game where each player can

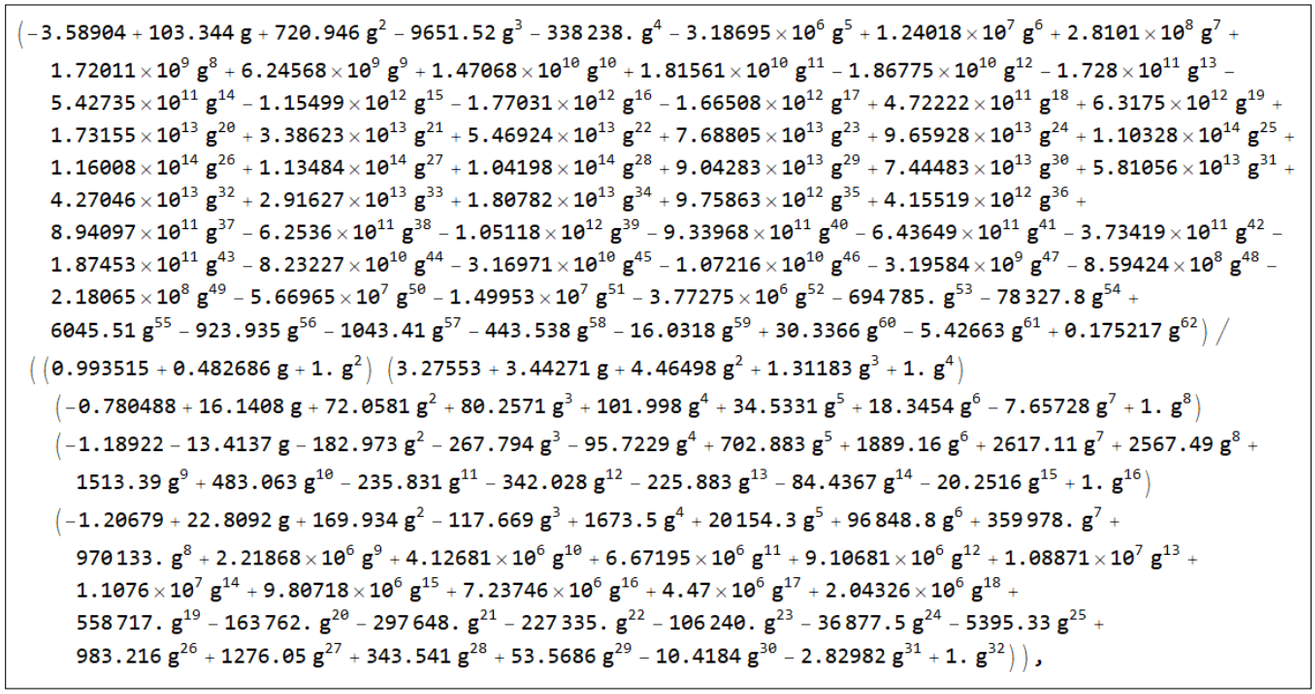

Figure 1: First entry of x^* after symbolic computations in Mathematica (g here stands for ①). Please observe how the provided solution is very difficult to read.

choose among 60 different strategies. This time, we filled the 60×60 payoff matrix by random G-scalars having 4 infinitesimal components. While Mathematica does not output the result in a reasonable amount of time for practical purposes, GM-S took only 122.487 seconds to compute the Nash Equilibrium, a quite remarkable result. For the sake of brevity, we omit both the matrix and the associated Nash equilibrium.

6. Possible Real-World Applications

In this section, we briefly propose some plausible examples in the fields of economics, politics and even engineering which seem to possess the non-Archimedean zero-sum property we discussed in this work. Once spotted, one could model them by means of GM and exploit GM-S algorithm we proposed here to study them in detail in the near future.

6.1. The Hotelling/Downs model

The Hotelling/Downs game of spatial competition [16, 21] has found uncountably many applications in economics and politics. Imagine two agents choosing their respective locations along a segment of finite length. If they are firms, locations may determine the degree of differentiation characterising their products, and therefore also affect market shares; if they are political parties, locations identify their electoral platforms, and determine political

consensus. If, in the original version belonging to industrial economics, prices are regulated, the Hotelling/Downs model is indeed a zero-sum game. Whenever the strategy space is discrete and finite, it ends up belonging to the set $\mathcal{ZG}$. What follows proposes two such examples, with a non-Archimedean interpretation.

Consider first electoral campaigns where only two parties (each with its own candidate) are involved and the linear segment which they are moving along is the range of all possible political platforms, from the extreme left to the extreme right. As soon as the set of political platforms is finite (e.g., is a unit segment), the problem belongs to $\mathcal{ZG}$ and, as stated by the minmax theorem, at least one Nash equilibrium exists. From Downs [16], we know that the game has a unique equilibrium with parties locating themselves at $1/2$, which is the position of the median voter. In such equilibrium, however, the electoral outcome is not determined because platforms are identical and therefore literally anything may happen. For instance, the left party may obtain the consensus of all voters, but still the right party may be aware that a sound opposition during the legislation is indeed valuable as it may determine the outcome at the next round. Whatever little a consolation this may represent, it is nonetheless true that having lost the elections has a value.

Now turn to the pristine interpretation of Hotelling's linear model as a duopoly game. The idea is that product differentiation serves the purpose of holding prices above average production costs, thereby boosting profits. As such, the model does not portray a zero-sum game because aggregate industry profits are not constant. However, the zero-sum property obtains as soon as one considers the scenario produced by price regulation. If price is exogenously given, then profits depend only on market shares, whose sum is constant. In such a case, the model replicates Downs' equilibrium precisely because market interaction has been nullified by the public authority, and analogous considerations hold. It is worth stressing that a situation like this is currently observable in Italy, where the price of masks has been regulated by the Government to ensure universal access to this basic safety tool under the pressure exerted by the CoViD19 pandemics.

6.2. Patrolling

Patrolling is a game theoretic model which involves two players: an attacker and a defender (or *patroller*). It is played on a finite graph, whose nodes are the object of the dispute between the players. The attacker aims to corrupt any node in the network, and to do it he/she needs an amount of time τ. The defender, in turn, has to find the path through the graph which minimizes the probability that a malicious agent will be able to successfully complete an attack without getting caught. Some famous application of such game are the security patrolling of museums or art galleries, the patrolling of a virtual network for malware, or the patrolling of a container yard or cargo warehouse.

Since when the defender prevents the attack he/she wins, the opponent looses and vice versa, such games fall in the set $\mathcal{ZG}$. Thus, one may wonder if a non-Archimedean extension of it is reasonable, especially because also the work in [10] is a zero-sum game on graphs and positively received the use of ① to study it. In order to affirmatively answer to the question, consider the problem of patrolling a museum where some exhibits are more precious than

others. The guardian path should take into account such additional information. In particular, the latter lets one divide the museum rooms by priority, such as the higher exhibits value is, the higher level of priority it has. Symmetrically, empty rooms, e.g., the entry room are assigned to the least level of priority. In this way, the gain or the loss of a player depends on the priority of the room that is attacked too. As soon as this priorities are lexicographic, the problem becomes a non-Archimedean one. In that case, representing each level of priority with different powers of ① is enough to properly represent and to numerically solve the game. For the sake of clarity, we proceed with an example. Assume to have only two levels of priority, high and low, represented by the powers 1 and 0, respectively. In such a case, the prevention of an attack to an high priority room corresponds to a gain of ① for the defender, while a successful incursion in a low priority room implies a loss of just 1 ($= ①^0$). Filling the payoff matrix with this logic lets one solve the non-Archimedean Patrolling problem by means of the extend GM-S algorithm proposed in this work.

6.3. Additional applications

This last part of the section gives glimpses of other possible applications without going into details. For instance, an infinitesimal quantity in economics may be the sentimental value attributed to the factory by the owner. In the case the firm goes out of business, from the owner perspective there is difference between maintaining or not the ownership of the facilities, even if the discrepancy of the two situations cannot be considered finite, i.e., economically relevant. Indeed, in both the cases there is not any production nor any economical income. However, the situation where the ownership is maintained can be represented as an infinitesimal surplus to the other case, since it can represent the sentimental value attributed by the owner to the factory, or even it may describe the potential incomes generated by the sell of the remaining facilities.

Another example of non-Archimedean zero-sum games in $\mathcal{NZG}$ is inspired by [25], where the zero-sum game theory is exploited for construction operation purposes. In particular, the author leveraged on it to evaluate which buildings construction was more convenient to invest in and to choose which realization strategy of infrastructure projects (such as the refurbishment of a damaged motorway or its expansion) maximized the quality-price ratio. A non-Archimedean version of such problems may be the one which adds a priority information over the quality criteria (gathering them in priority levels, similarly to what suggested for Patrolling games), or the one which splits each criterion in multiple sub-criteria. For instance, consider the quality improvement of a building due to the presence of heating. If it lacks the improvement is 0, while if it is present the improvement is 1. However, there exist several kind of heating, such as gas or district heating. Preferring the latter to the former, a proposal could be to represent the improvement due to district heating with $1 + ①^{-1}$, while the one due to gas heating just by 1 ($= 1 + 0①^{-1}$). The advantages are twofold. Firstly, one is able to embed in the problem that the priority is a house with heating, regardless the kind. Then, the modeler does not have to care about how more important the gas and the district heating are with respect to the absence of heating. Indeed, this settings would have took into account the other criteria too, in order to avoid a hegemonic role of the district heating on the overall decision process.

7. Conclusions

In this note we have introduced for the first time non-Archimedean zero-sum games. Then, we have designed and implemented GM-S, a non-Archimedean Simplex able to find one Nash equilibrium of a non-Archimedean zero-sum game leveraging on Grossone Methodology. After providing four numerical illustrations, obtained by numeric simulation performed using an Infinity Computer simulator, we have discussed possible real-world applications of this new modeling tool. Being our tool numeric and not symbolic, we have shown how it is able to solve even high-dimensional non-Archimedean zero-sum games. Finally, we remark how results obtained in this study would not have been attained without our previous achievements in non-Archimedean linear programming [5, 7, 8] and non-Archimedean Prisoner's Dilemmas [18, 19, 20]. As a future work, we will delve into the details of the applications sketched here, along with quantitative experiments.

Appendix A. The 8x8 non-Archimedean zero-sum game

This appendix contains some data related to the experiment discussed in Section 5.4 about the 8×8 non-Archimedean zero-sum game. In Table A.1 we reported the payoff matrix. For space reasons, we had to round all the entries components up to the second decimal digit and to split the matrix on two pieces, the rightmost and leftmost ones. In Table A.3 and Table A.2 instead, we disclosed GM-S iterations to solve the problem. Again for space reasons, we reported only the non-zero entries of the vertexes visited by the algorithm.

Table A.1: 8x8 payoff matrix of the game presented in Section 5.4

$0.8 - 0.21①^{-1} - 0.84①^{-2}$	$0.69 - 0.26①^{-1} - 0.02①^{-2}$	$0.13 + 0.4①^{-1} - 0.97①^{-2}$	$0.0 - 0.92①^{-1} - 0.39①^{-2}$	$0.52 + 0.97①^{-1} + 0.59①^{-2}$	$0.84 - 0.77①^{-1} - 0.5①^{-2}$	$0.31 - 0.26①^{-1} - 0.15①^{-2}$	$0.75 - 0.7①^{-1} + 0.7①^{-2}$
$0.21 - 0.81①^{-1} - 0.53①^{-2}$	$0.17 - 0.01①^{-1} - 0.83①^{-2}$	$0.59 - 0.96①^{-1} + 0.73①^{-2}$	$0.7 - 0.08①^{-1} - 0.0①^{-2}$	$0.1 - 0.23①^{-1} + 0.71①^{-2}$	$0.53 - 0.44①^{-1} + 0.43①^{-2}$	$0.25 - 0.68①^{-1} + 0.52①^{-2}$	$0.97 + 0.46①^{-1} - 0.3①^{-2}$
$0.31 - 0.34①^{-1} + 0.75①^{-2}$	$0.74 + 0.78①^{-1} + 0.27①^{-2}$	$0.23 - 0.56①^{-1} - 0.03①^{-2}$	$0.56 + 0.3①^{-1} - 0.19①^{-2}$	$0.59 - 0.08①^{-1} - 0.03①^{-2}$	$0.02 - 0.3①^{-1} + 0.28①^{-2}$	$0.78 - 0.79①^{-1} + 0.04①^{-2}$	$0.09 - 0.01①^{-1} + 0.42①^{-2}$
$0.48 + 0.02①^{-1} - 0.46①^{-2}$	$0.38 - 0.6①^{-1} + 0.2①^{-2}$	$0.21 + 0.72①^{-1} - 0.61①^{-2}$	$0.74 + 0.89①^{-1} + 0.13①^{-2}$	$0.59 + 0.68①^{-1} + 0.39①^{-2}$	$0.62 - 0.49①^{-1} + 0.67①^{-2}$	$0.35 - 0.67①^{-1} - 0.98①^{-2}$	$0.92 + 0.35①^{-1} + 0.51①^{-2}$
$0.37 - 0.72①^{-1} - 0.09①^{-2}$	$0.2 - 0.22①^{-1} + 0.14①^{-2}$	$0.82 - 0.81①^{-1} - 0.25①^{-2}$	$0.78 - 0.65①^{-1} - 0.81①^{-2}$	$0.49 - 0.58①^{-1} - 0.42①^{-2}$	$0.22 + 0.25①^{-1} + 0.38①^{-2}$	$0.12 + 0.87①^{-1} - 0.76①^{-2}$	$0.53 - 0.95①^{-1} + 0.2①^{-2}$
$0.62 + 0.59①^{-1} - 0.31①^{-2}$	$0.85 - 0.53①^{-1} - 0.99①^{-2}$	$0.04 - 0.78①^{-1} + 0.81①^{-2}$	$0.66 + 0.7①^{-1} + 0.0①^{-2}$	$0.27 - 0.77①^{-1} + 0.74①^{-2}$	$0.59 + 0.43①^{-1} + 0.79①^{-2}$	$0.97 + 0.63①^{-1} + 0.8①^{-2}$	$0.2 + 0.87①^{-1} - 0.16①^{-2}$
$0.93 + 0.56①^{-1} - 0.36①^{-2}$	$0.3 + 0.65①^{-1} - 0.35①^{-2}$	$0.4 + 0.91①^{-1} + 0.51①^{-2}$	$0.04 - 0.71①^{-1} - 0.58①^{-2}$	$0.86 - 0.19①^{-1} - 0.02①^{-2}$	$1.0 - 0.62①^{-1} + 0.03①^{-2}$	$0.79 - 0.57①^{-1} + 0.61①^{-2}$	$0.41 + 0.05①^{-1} + 0.69①^{-2}$
$0.01 - 0.54①^{-1} + 0.71①^{-2}$	$0.62 + 0.18①^{-1} + 0.38①^{-2}$	$0.42 + 0.17①^{-1} + 0.78①^{-2}$	$0.29 - 0.39①^{-1} + 0.17①^{-2}$	$0.46 - 0.87①^{-1} - 0.42①^{-2}$	$0.55 + 0.27①^{-1} - 0.42①^{-2}$	$0.61 + 0.5①^{-1} - 0.13①^{-2}$	$0.92 + 0.72①^{-1} + 0.53①^{-2}$

Table A.2: GM-S iterations to solve the 8×8 problem: basic solutions

Iter.	$\mathbf{x}^*$
1	$[1.0①^0,\ 1.0①^0,\ 1.0①^0,\ 1.0①^0,\ 1.0①^0,\ 1.0①^0,\ 1.0①^0,\ 1.0①^0,\ ]$
2	$[1.03 + 0.34①^{-1} + 0.37①^{-2},\ 0.29 + 0.05①^{-1} + 0.57①^{-2},\ 0.74 + 1.34①^{-1} + 0.62①^{-2},\ 0.21 - 0.2①^{-1} - 0.33①^{-2},\ 0.55 + 0.79①^{-1} - 0.08①^{-2},\ 0.58 + 0.41①^{-1} - 0.03①^{-2},\ 0.77 - 0.25①^{-1} - 0.13①^{-2},\ 0.07 + 0.32①^{-1} + 0.26①^{-2}]$
3	$[1.1 + 0.74①^{-1} + 0.97①^{-2},\ 0.24 - 0.21①^{-1} + 0.33①^{-2},\ 0.72 + 1.35①^{-1} + 1.05①^{-2},\ 0.15 - 0.51①^{-1} - 0.77①^{-2},\ 0.52 + 0.69①^{-1} + 0.01①^{-2},\ 0.54 + 0.29①^{-1} - 0.07①^{-2},\ 0.76 - 0.36①^{-1} - 0.33①^{-2},\ 0.08 + 0.37①^{-1} + 0.43①^{-2}]$
4	$[0.31 - 0.68①^{-1} + 1.11①^{-2},\ 1.09 + 1.13①^{-1} + 0.26①^{-2},\ 0.47 + 1.99①^{-1} + 0.18①^{-2},\ 0.04 - 0.63①^{-1} - 0.2①^{-2},\ 0.43 + 0.88①^{-1} - 0.03①^{-2},\ 0.27 + 0.85①^{-1} - 0.61①^{-2},\ 0.53 + 0.05①^{-1} - 0.88①^{-2},\ 0.21 + 0.38①^{-1} + 0.13①^{-2}]$
5	$[0.41 - 2.66①^{-1} + 7.18①^{-2},\ 1.09 + 1.31①^{-1} - 2.24①^{-2},\ 0.39 + 3.63①^{-1} - 5.23①^{-2},\ 0.4 + 1.44①^{-1} - 1.66①^{-2},\ 0.19 + 2.58①^{-1} - 5.61①^{-2},\ 0.46 + 1.49①^{-1} - 4.74①^{-2},\ 0.07 - 1.42①^{-1} + 2.7①^{-2},\ 0.25 - 0.38①^{-1} + 0.78①^{-2}]$
6	$[0.62 + 0.25①^{-1} + 0.69①^{-2},\ 1.08 + 1.46①^{-1} + 1.48①^{-2},\ 0.21 + 1.31①^{-1} + 0.96①^{-2},\ 0.34 + 0.61①^{-1} + 0.35①^{-2},\ 0.3 - 0.73①^{-1} - 0.8①^{-2},\ 0.24 + 1.02①^{-1} + 0.79①^{-2},\ 0.08 + 1.37①^{-1} + 1.06①^{-2},\ 0.34 + 1.08①^{-1} + 0.74①^{-2}]$
7	$[0.88 + 1.94①^{-1} + 2.64①^{-2},\ 1.07 + 1.71①^{-1} + 3.51①^{-2},\ 0.27 + 0.13①^{-1} + 0.02①^{-2},\ 0.11 - 2.07①^{-1} - 2.72①^{-2},\ 0.44 + 2.59①^{-1} + 4.14①^{-2},\ 0.18 + 2.33①^{-1} + 3.78①^{-2},\ 0.23 + 1.52①^{-1} + 1.61①^{-2},\ 0.46 + 2.06①^{-1} + 3.13①^{-2}]$
8	$[1.02 - 0.91①^{-1} + 0.2①^{-2},\ 1.07 + 1.97①^{-1} + 0.31①^{-2},\ 0.23 + 0.94①^{-1} + 0.85①^{-2},\ 0.55 + 0.55①^{-1} - 0.54①^{-2},\ 0.12 - 2.37①^{-1} - 1.33①^{-2},\ 0.23 + 1.43①^{-1} - 0.75①^{-2},\ 0.36 - 0.97①^{-1} - 1.03①^{-2},\ 0.52 + 0.97①^{-1} - 0.75①^{-2}]$
9	$[0.99 - 0.94①^{-1} + 0.55①^{-2},\ 0.84 + 1.4①^{-1} + 1.01①^{-2},\ 0.35 + 0.98①^{-1} - 0.37①^{-2},\ 0.39 - 0.15①^{-1} - 0.52①^{-2},\ 0.28 - 1.95①^{-1} - 1.39①^{-2},\ 0.26 + 1.04①^{-1} - 1.93①^{-2},\ 0.28 - 1.11①^{-1} - 0.55①^{-2},\ 0.28 + 0.48①^{-1} + 0.35①^{-2}]$
10	$[0.57 + 0.27①^{-1} + 0.43①^{-2},\ 0.54 + 1.17①^{-1} + 2.51①^{-2},\ 0.4 + 0.0①^{-1} - 0.77①^{-2},\ 0.53 + 0.15①^{-1} - 0.68①^{-2},\ 0.11 + 0.42①^{-1} + 1.29①^{-2},\ 0.29 - 1.3①^{-1} - 1.56①^{-2},\ 0.03 + 0.53①^{-1} + 0.85①^{-2},\ 0.1 + 0.65①^{-1} + 1.12①^{-2}]$
11	$[0.32 - 1.18①^{-1} - 4.48①^{-2},\ 0.41 + 1.7①^{-1} + 0.39①^{-2},\ 0.72 + 0.61①^{-1} + 2.58①^{-2},\ 0.32 - 0.48①^{-1} + 0.05①^{-2},\ 0.18 + 1.32①^{-1} + 4.39①^{-2},\ 0.1 + 0.88①^{-1} + 0.68①^{-2},\ 0.16 + 0.39①^{-1} + 0.61①^{-2},\ 0.19 + 0.29①^{-1} + 1.46①^{-2}]$
12	$[0.34 - 0.37①^{-1} + 1.62①^{-2},\ 0.3 + 0.31①^{-1} - 3.6①^{-2},\ 0.5 - 0.82①^{-1} + 1.72①^{-2},\ 0.44 + 1.0①^{-1} + 4.91①^{-2},\ 0.16 + 0.85①^{-1} - 1.1①^{-2},\ 0.21 + 0.85①^{-1} - 2.31①^{-2},\ 0.16 + 0.2①^{-1} - 0.66①^{-2},\ 0.09 - 0.29①^{-1} + 1.7①^{-2}]$

Table A.3: GM-S iterations to solve the 8×8 problem: basis and objective function

Iter.	**Base**	$\tilde{\mathbf{c}}^T\mathbf{x}^*$
1	{9, 10, 11, 12, 13, 14, 15, 16}	$-8.0①^1$
2	{9, 10, 11, 12, 13, 3, 15, 16}	$-3.22①^1 - 3.48 - 1.21①^{-1}$
3	{9, 10, 11, 12, 13, 3, 15, 22}	$-2.93①^1 - 2.36 - 0.96①^{-1}$
4	{2, 10, 11, 12, 13, 3, 15, 22}	$-1.73①^1 - 4.54 + 1.1①^{-1}$
5	{2, 10, 17, 12, 13, 3, 15, 22}	$-1.44①^1 - 10.64 + 18.61①^{-1}$
6	{2, 10, 17, 12, 19, 3, 15, 22}	$-0.85①^1 - 2.9 - 2.21①^{-1}$
7	{2, 21, 17, 12, 19, 3, 15, 22}	$-0.37①^1 - 0.02 - 0.95①^{-1}$
8	{2, 21, 17, 12, 19, 3, 18, 22}	$-0.23①^1 - 3.03 - 1.9①^{-1}$
9	{2, 21, 17, 7, 19, 3, 18, 22}	$-2.18 - 1.44①^{-1} - 1.19①^{-2}$
10	{2, 5, 17, 7, 19, 3, 18, 22}	$-2.03 - 1.59①^{-1} - 1.5①^{-2}$
11	{2, 5, 17, 7, 19, 3, 1, 22}	$-1.95 - 1.96①^{-1} - 2.92①^{-2}$
12	{2, 5, 6, 7, 19, 3, 1, 22}	$-1.95 - 1.83①^{-1} - 1.24①^{-2}$

Acknowledgements

This work has been partially supported by the University of Pisa funded project PRA_2018_81 "Wearable sensor systems: personalized analysis and data security in healthcare" and partially by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

References

[1] Ilan Adler. The equivalence of linear programs and zero-sum games. *International Journal of Game Theory*, 42(1):165–177, 2013.

[2] Mokhtar S. Bazaraa, John J. Jarvis, and Hanif D. Sherali. *Linear Programming and Network Flows*. John Wiley & Sons, 1990.

[3] Siegfried Bosch, Ulrich Güntzer, and Reinhold Remmert. *Non-Archimedean Analysis: A Systematic Approach to Rigid Analytic Geometry*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2012.

[4] Tullio Levi Civita. *Sui numeri transfiniti*. Tipografia della R. Accademia dei Lincei, 1898.

[5] Marco Cococcioni, Alessandro Cudazzo, Massimo Pappalardo, and Yaroslav D. Sergeyev. Grossone methodology for lexicographic mixed-integer linear programming problems. In Yaroslav D. Sergeyev and Dmitri E. Kvasov, editors, *Numerical Computations: Theory and Algorithms*, pages 337–345, Cham, 2020. Springer International Publishing.

[6] Marco Cococcioni, Alessandro Cudazzo, Massimo Pappalardo, and Yaroslav D Sergeyev. Solving the lexicographic multi-objective mixed-integer linear programming problem using branch-and-bound and grossone methodology. *Comm. in Nonlinear Science and Numerical Simulation*, 84:105177, 2020.

[7] Marco Cococcioni and Lorenzo Fiaschi. The Big-M method with the numerical infinite M. *Optimization Letters*, 2020, doi:10.1007/s11590-020-01644-6.

[8] Marco Cococcioni, Massimo Pappalardo, and Yaroslav D. Sergeyev. Lexicographic multi-objective linear programming using grossone methodology: Theory and algorithm. *Applied Mathematics and Computation*, 318:298 – 311, 2018.

[9] Sonia De Cosmis and Renato De Leone. The use of grossone in mathematical programming and operations research. *Applied Mathematics and Computation*, 218(16):8029–8038, 2012.

[10] Louis D'Alotto. Infinite games on finite graphs using grossone. In Yaroslav D. Sergeyev and Dmitri E. Kvasov, editors, *Numerical Computations: Theory and Algorithms*, pages 346–353, Cham, 2020. Springer International Publishing.

[11] George B. Dantzig. Programming in a linear structure. Technical report, United Air Force, Washington, D.C., 1948.

[12] George B. Dantzig and Mukund N. Thapa. *Linear Programming 1: Introduction*. Springer-Verlag, New York, 1997.

[13] George B. Dantzig and Mukund N. Thapa. *Linear Programming 2: Theory and Extensions*. Springer-Verlag, New York, 2003.

[14] Renato De Leone, Nadaniela Egidi, and Lorella Fatone. The use of grossone in elastic net regularization and sparse support vector machines. *Soft Computing*, pages 1–9, 2020.

[15] Renato De Leone, Giovanni Fasano, Massimo Roma, and Yaroslav D Sergeyev. Iterative grossone-based computation of negative curvature directions in large-scale optimization. *Journal of Optimization Theory and Applications*, 186(2):554–589, 2020.

[16] Anthony Downs. *An Economic Theory of Democracy*. Harper and Row, New York, 1957.

[17] Alberto Falcone, Alfredo Garro, Marat S. Mukhametzhanov, and Yaroslav D. Sergeyev. A simulink-based infinity computer simulator and some applications. In Yaroslav D. Sergeyev and Dmitri E. Kvasov, editors, *Numerical Computations: Theory and Algorithms*, pages 362–369, Cham, 2020. Springer International Publishing.

[18] Lorenzo Fiaschi and Marco Cococcioni. Numerical Asymptotic Results in Game Theory using Sergeyev's Arithmetic of Infinity. *Int. Journal on Unconventional Computing*, 14:1–25, 2018.

[19] Lorenzo Fiaschi and Marco Cococcioni. Generalizing pure and impure iterated prisoner's dilemmas to the case of infinite and infinitesimal quantities. In Yaroslav D. Sergeyev and Dmitri E. Kvasov, editors, *Numerical Computations: Theory and Algorithms*, pages 370–377, Cham, 2020. Springer International Publishing.

[20] Lorenzo Fiaschi and Marco Cococcioni. Non-Archimedean Game Theory: A Numerical Approach. *Applied Mathematics and Computation*, 2020, doi:10.1016/j.amc.2020.125356.

[21] Harold Hotelling. Stability in Competition. *The Economic Journal*, 39:41–57, 1929.

[22] Tullio Levi-Civita. *Sugli infiniti ed infinitesimi attuali, quali elementi analitici*. Tip. Ferrari, 1893.

[23] Roger B Myerson. *Game Theory*. Harvard university press, 2013.

[24] Guillermo Owen. *Game Theory*. Academic Press, 1995.

[25] Friedel Peldschus. Experience of the game theory application in construction management. *Technological and Economic Development of Economy*, 14(4):531–545, 2008.

[26] Davide Rizza. Numerical methods for infinite decision-making processes. *International Journal of Unconventional Computing*, 14(2):139–158, 2019.

[27] Abraham Robinson. *Non-standard analysis*. Princeton University Press, 2016.

[28] Peter Schneider. *Nonarchimedean functional analysis*. Springer Science & Business Media, 2013.

[29] Yaroslav D. Sergeyev. *Computer system for storing infinite, infinitesimal, and finite quantities and executing arithmetical operations with them*. USA patent 7,860,914, 2010.

[30] Yaroslav D. Sergeyev. The olympic medals ranks, lexicographic ordering and numerical infinities. *Math. Intelligencer*, 37:4–8, 2015.

[31] Yaroslav. D. Sergeyev. Numerical infinities and infinitesimals: Methodology, applications, and repercussions on two Hilbert problems. *EMS Surveys in Mathematical Sciences*, 4:219–320, 2017.

[32] Yaroslav D. Sergeyev. Independence of the grossone-based infinity methodology from non-standard analysis and comments upon logical fallacies in some texts asserting the opposite. *Foundations of Science*, 24(1):153–170, Mar 2019.

[33] Lloyd Stowell Shapley and RN Snow. Basic solutions of discrete games. *Annals of Mathematics Studies*, 1950.

[34] Majid Soleimani-Damaneh. Modified Big-M method to recognize the infeasibility of linear programming models. *Knowledge-Based Systems*, 21(5):377 – 382, 2008.

[35] Robert J Vanderbei et al. *Linear programming*. Springer, 2015.

[36] Alan R Washburn. *Two-person zero-sum games*. Springer, 2014.