CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Cybernetics

Software or Research Project

Solving Continuous Games With Oracle Algorithms

Bc. Tomáš Kasl
kasltoma@fel.cvut.cz

January 2024
Supervisor: doc. Ing. Tomáš Kroupa, Ph.D

/ Declaration

I swear I have worked on this project, and its experiments, with honesty, and on my own.

Prague, 15 Jan 2024

Abstrakt / Abstract

Řešení spojitých her s kompaktními prostory strategií představuje velkou výzvu pro současnou teorii her. Hledání smíšených ekvilibrií ve dvouhráčových modelech s nulovým součtem je obtížně řešitelný problém, k jehož problému se často používají numerické iterační metody založené na různých oracle přístupech (výpočet best response atd.) Nový článek [2] je založen na jednom takovém přístupu, jehož varianta byla již kdysi použita v práci [1]. Tato práce se soustředí na porovnání efektivity obou algoritmů pomocí experimentů, a na experimentální ověření tvrzení o výpočetní náročnosti uvedené v původním článku [1].

Klíčová slova: hry, teorie her, spojité hry, nekonečné hry, algoritmus Double Oracle, algoritmus Expectated Regret Minimization

Solving continuous games with compact action spaces presents a huge challenge for the current game theory. Searching mixed equilibria in two-game models with zero some is fairly difficult problem, usually done by numeric iterative methods based on various approaches to oracles (i.e. computation of the best response etc.). New paper [1] is based on one such approach, a special variant of which was already used in work [2] before. This work focuses on comparison of effectiveness of these algorithms by experiments, and experimental verification of claims on computation complexity given in the source paper [1].

Keywords: games, game theory, continuous games, infinite games, Double oracle algorithm, ERM algorithm, Expected Regret Minimization

/ Contents

Chapter 1
Introduction

1.1 Games & Game Theory

Game theory is a mathematical discipline, which tries to model strategic interactions between multiple agents (often called players) with varying degrees of rationality in a shared environment, such as various social or economic situations, conflicts, and also games[3].

In the games, every participating player aims to maximize his individual gain from the game among the individuals. That means every player strives to find his own best (optimal) strategy, which consequently depends on the strategies of other players.

The setting of two-player zero-sum games with finite action spaces is quite well understood. As the *solution* of the game is often called a Nash Equilibrium (NE)[4]. An NE is a stable state of the game, where no player has any incentive to deviate from the strategy he has already adopted.

While finding an NE in this setting is a solved problem (very well explained in the textbook Multiagent Systems[3]), finding an NE in infinite (also called continuous) games is not fully understood yet. Since no algorithm to compute an NE is known, iterative *oracle*-based algorithms converging to an ϵ-NE are usually used.

By allowing for some ϵ error, we can get away with modeling the strategies in an NE by a discrete probability distribution, while still getting empirically precise models for real-world usage. By *oracles* are meant blackbox-like optimizers used, for example, to find the best response (from the whole continuum of his possible actions) of an agent against his opponent.

The ability to find an NE in the infinite two-player zero-sum game is important, because not only does it provide a solution to problems that we can model in this framework, but can be used as a part of solving more complex problems.

1.1.1 Notation

A two-player zero-sum game is defined as:

$$G = (X, Y, u)$$

where:

- X is the action space for Player 1, it can be a set of actions x_i, or a continuum.
- Y is the action space for Player 2, it can be a set of actions y_j, or a continuum.
- $u : X \times Y \to R$ - the utility function giving the first player's reward/cost for the chosen actions.

Notes:

- X can be the same as Y, but it is not a requirement

- It is enough to model only one utility function u, since $u_1(x_i, y_j) = -u_2(x_i, y_j)$

A (mixed) strategy p is a probability distribution over the player's action space. A pure strategy is a special case of a mixed strategy.

When a strategy is played, the final action is i.i.d drawn from the probability distribution, after all the players have chosen their strategies.

The expected utility gain for playing mixed strategies over finite action spaces is

$$U(p, q) = \sum_{x_i \in X} \sum_{y_j \in Y} u(x_i, y_j) \cdot p_i \cdot q_j$$

where p and q are the respective strategies for both players.

A subgame of $G(X, Y, u)$ is $G'(A, B, u)$, where $A \subseteq X$ and $B \subseteq Y$, that means a game, where players have restricted their action spaces. In this work, we only encounter subgames with finite action spaces (lists of actions).

A pair of strategies (p^*, q^*) is a Nash equilibrium in a game G if

$$U(p, q^*) \leq U(p^*, q^*) \leq U(p^*, q)$$

holds for any p and any q over X and Y.

1.2 Solving games in normal form

1.2.1 Linear Program

For two-player zero-sum games, where each player has a finite list of possible actions, from which they can choose, the solution is well known. We can enumerate the actions of both players ($a_j \in A$ and $b_k \in B$), create a matrix M consisting of values of u of all the pairs of actions, and solve the following linear program:

Primal task:		Dual task:	
maximize	x_0	minimize	y_0
subject to:	$M^T\mathtt{x} - \mathtt{1}x_0 \geq 0$	subject to:	$M\mathtt{y} - \mathtt{1}y_0 \leq 0$
	$\sum_{a_j \in A} x_j = 1$		$\sum_{b_k \in B} y_k = 1$
	$\mathtt{x} \geq 0$		$\mathtt{y} \geq 0$

1.3 Solving games with infinite action spaces

The matrix representation of the game is no longer possible, and so the LP cannot be directly used for obtaining the NE anymore. As we settle for finding an ϵ-NE, we can obtain a solution as a pair of strategies with finite supports.

To compute the ϵ-NE, then, iterative algorithms are used, which aim to construct it by picking the *good* actions from the continuum, one *bestResponse* (one type of *oracle*) at a time, and then using the aforementioned linear program (another type of *oracle*) to solve the subgame of current iteration.

Usually, finding the *bestResponse* means searching for a global optimum of a function, basically without constraints on the function's shape. That is a very complex problem of its own, and it is a focus of this project.

Also note that in the algorithm definitions (and elsewhere), I have modified the notations compared to their sources, so that it is consistent in this document.

1.3.1 Methods

The methods worth mentioning for this work are:

- Fictitious play - estimates the equilibrial strategies as the average over *many bestResponses*. Not experimented with in this project.[5]
- Double oracle (DO)[2]
- Expected Regret Minimization (ERM)[1]

1.3.2 Double Oracle[2]

This algorithm aims to achieve an ϵ-NE by adding 1 action for each of the players at the time, such that it is the best response to the currently reached outcome, and therefore to continually converge to an NE. In each iteration, for each of the players, the best response is found, and NE for the newly formed subgame is solved.

The algorithm was designed to find the ϵ-NE of huge matrix games by solving the LP over much smaller sub-game matrices (and therefore offering faster compute time)[6]. Later, it was generalized into the setting of infinite action spaces [2]

```
algorithm 1:
DOUBLE ORACLE
```

inputs: game $G = (X, Y, u)$, number ϵ
output: equilibrial strategies p_t^*, q_t^*

1. $A_0 \leftarrow \{a\}, B_0 \leftarrow \{b\}$, where $a \in X$ and $b \in Y$ are arbitrary actions
2. for $t = 1, ...$
   (a) find NE (p_t^*, q_t^*) of subgame (A_t, B_t, u), (A_t, B_t are finite sets in time t)
   (b) $a \leftarrow bestResponse(X, q_t^*)$; b $\leftarrow bestResponse(Y, p_t^*)$
   (c) $A_{t+1} \leftarrow A_t \cup \{a\}$; $B_{t+1} \leftarrow B_t \cup \{b\}$
   (d) if $u(p_t^*, b) - u(a, q_t^*) \leq \epsilon$:
   Return (p_t^*, q_t^*)

Notice the algorithm definition does not specify, how the *bestResponse* or finding the NE of a subgame is implemented. They are black box-like *oracles*.

This algorithm is proven to converge to an ϵ-NE in a finite number of steps for games, where the action spaces are either finite or hypercube continuums (e.g. $[0, 1]^n$, but other boundary values are also allowed).

1.3.3 Expected Regret Minimization[1]

This is a generalization of the general ERM approach used for online learning of strategy-making, where an agent needs to act in an environment with as much flexibility as possible, into the setting of two-player games, where each player's environment is also defined by the other player, presented in a recent paper[1]. By *online learning* is meant the agent's ability to act within the environment even before the learning is finished.

The algorithm is composed of a main routine, and a subroutine, and besides setting a constant *epsilon*, it also requires a *magic constant* C, which I will show to cause some concerns.

EXPECTED REGRET MINIMIZATION

algorithm 2:
epsilon-approximate Nash Equilibrium for a zero-sum game (main routine)

inputs: game $G = (X, Y, u)$, number ϵ, number C
output: equilibrial strategies p_t^*, q_t^*

1. $A_0 \leftarrow \{a\}, B_0 \leftarrow \{b\}$, where $a \in X$ and $b \in Y$ are arbitrary actions
2. For $t = 1, 2, ...$
   (a) $(responses_a,\ probabilities_b) \leftarrow NASH(G, B_{t-1}, \epsilon)$
   (b) $A_t \leftarrow A_{t-1} \cup responses_a$
   (c) $(responses_b,\ probabilities_a) \leftarrow NASH(G, A_t, \epsilon)$
   (d) $B_t \leftarrow B_{t-1} \cup responses_b$
   (e) if $\text{Val}(A_t, B_{t-1}) \geq \text{Val}(A_{t-1}, B_{t-1}) - \epsilon$ or $\text{Val}(A_t, B_t) \leq \text{Val}(A_t, B_{t-1}) + \epsilon$:
   Return NE (p_t^*, q_t^*) of subgame G'=(A_t, B_t, u)

The $Val(A, B)$ oracle finds the NE for a subgame and returns the utility value in equilibrial strategies.

NASH, algorithm 3:
epsilon Nash Equilibrium for a half-infinite zero-sum game (subroutine)

inputs: game $G = (X, Y, u)$, actions space A_t, number ϵ
output: equilibrial strategies p_t^*, q_t^*

1. $V \leftarrow \left\lceil \frac{C log\|A_t\|}{\epsilon^2} \right\rceil$; $\eta \leftarrow \sqrt{\frac{log\|A_t\|}{2V}}$
2. $p^1 = (p_1^1, ..., p_n^1)$ is a uniform distribution over A_t
3. $responses \leftarrow bestResponse(Y, p_1)$
4. for $v = 2, ..., V$
   (a) for $i = 1, ..., N$: $p_i^v = p_i^{v-1} exp(\eta\ u(a_i, b_{v-1}))/Z^v$
   where:
   $$Z^v = \sum_{j=1}^{n} p_j^{v-1} exp(\eta\ u(a_j, b_{v-1}))$$
   (b) $b_v \leftarrow bestResponse(A_t, p_v)$; $responses \leftarrow responses \cup b_v$
5. Return $(responses,\ avg(p^i))$

The version of algorithm 3 for the other player is defined analogously.

This implementation of algorithm 3, which uses the *bestResponse* oracle and multiplicative weight update (MWU), is only a proposal given in the source paper. Other algorithms conforming to some given assumptions may be viable.

Chapter 2
Project

2.1 Project

2.1.1 Goals of the project

Finding Nash Equilibria in the setting of continuous/infinite games is not yet solved. While the somewhat straightforward Fictitious Play has all the important convergence guarantees[7], its convergence speed is often too slow for practical use. This project aims to examine the ERM algorithm, and focuses on 2 main experiments: 1. Comparison of convergence properties between Double Oracle and Expected Regret Minimization algorithms on games, where players' action spaces are hypercubes, and 2. Practical confirmations of some (complexity/space) bounds claimed by the ERM[1] paper's authors, by experiments. In a sense, this can be viewed as a testing of some of the proposals and claims given in the source paper.

2.1.2 Format of games

A game is defined by

- X: a hypercube of action space of Player 1 ($[a, b] \times [c, d], \times ... \times [g, h]; a, .., h \in R$)
- Y: a hypercube of action space of Player 2 ($[r, s] \times [t, u], \times ... \times [v, w]; r, .., w \in R$)
- u: X $\times Y \rightarrow R$, the utility function

Also, we have the definitions of utility functions in mixed strategies, where

- Given an action a, action list B and their weights q, utility is equal to $\sum_{b_i \in B} u(a, b_i) \cdot q_i$
- Given an action b, action list A and their weights p, utility is equal to $\sum_{a_i \in A} u(a_i, b) \cdot p_i$
- Given an action list a, action list b and their weights p and q, utility is equal to $\sum_{a_i \in A} \mathrm{U} = \sum_{b_j \in B} u(a_i, b_j) \cdot p_i \cdot q_j$

2.1.3 Oracles

Represent `black boxes` of optimizations. The Oracles used are

- Best Response Oracles
  - Player 1's *bestResponse* Oracle: given a mixed strategy of Player 2, find the best response (single action $x \in X$)
  - Player 2's *bestResponse* Oracle: given a mixed strategy of Player 1, find the best response (single action $y \in Y$)

  Since the utility function is a n-variable real-valued function, for the experiments, I am using the *Powell's method*[8] of finding the global optima, provided by the SciPy library. Since I am using it for various kinds of utility functions, I use it as a black box, which, unfortunately, does not guarantee to find the actual global optima, and might be a source of noise in the experiments' outcomes.

- *gameValue* Oracle: Computes the value of the game at a Nash Equilibrium, given a finite set of actions for both players, using the linear program stated in section 1.2.1. The LP is used for finding the NE of all subgames in this project.

2.1.4 Double Oracle and ERM implementation notes

The implementation is available on GitHub[1]. The code can be started by running *main.py*, or by inspecting the *notebook.ipynb*.

While the algorithms never specify, which action should be selected as the very first one, I create the initial set of strategies as all the corners of the hypercube defining their respective player's action space. Doing this reduces randomness and is beneficial to the reproducibility of the experiments.

The *Powell's method* of searching the global optima requires a starting point as a parameter. Since the *bestResponse* oracle is called *often*, the starting point is chosen randomly, which decreases the probability the actual global optimum (and therefore, the actual best response) is never reached. Versions of software used are: Python 3.11.6, SciPy 1.11.3 and NumPy 1.25.2.

2.2 Experiments and Outcomes

2.2.1 Games

Games are constructed using multiple functions found in the literature or research papers, all their definitions are found in the *funcs.py* file, where a link to their *.pdf* source can be found. Generally, all the games are based on n-variable real functions, where n is an even number (so it is divisible by the 2 players), and the players' action spaces are hypercubes.

The list has been mostly taken from a colleague's work[9]. The perhaps unreasonably good performance of the ERM algorithm is caused by the fact it searches for any mixed strategies leading to $\epsilon-$NE. On the contrary, strictly pure strategies are mostly searched for in the papers, from where these functions are taken from. Also, the pure strategies do not necessarily compose an NE, the value in the computed mixed NE can be different.

2.2.2 Convergence comparison, part 1

The implementation of functions used is available in *funcs.py*[2]

For all of the functions, the comparison between the convergence of the 2 algorithms is visualized on 2 plots. The first one shows the reached lower and upper bounds admissible by the players, in each iteration, found by each of the 2 algorithms. The second plot is very similar, but has a different horizontal axis. Instead of iterations, the axis is now based on the number of *bestResponse* oracle calls.

Because the number of oracle calls changes in each iteration, this is not really a rigid exact plot, but it is sufficient to put the tradeoff between a number of iterations and a number of oracle calls into perspective.

The parameters used for the specific algorithm run are in the plots' titles.

[1] https://github.com/grifun/ERMvsDO
[2] https://github.com/grifun/ERMvsDO/blob/main/code/funcs.py

1. An ad-hoc function: just a polynomial of second degree

$$u(x, y) = 5xy - 2x^2 - 2xy^2 - y$$

where:

$$X = Y = [-1, 1]$$

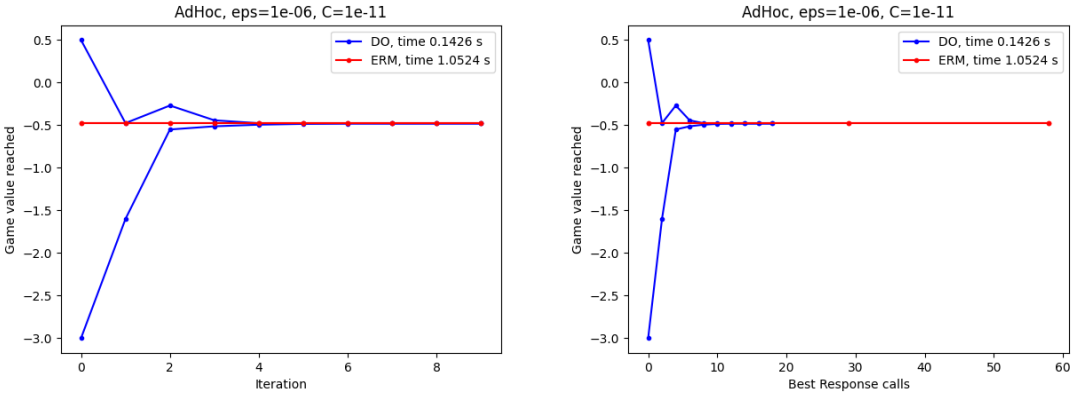

Figure 2.1. Converge comparison on an ad-hoc function.

Generally, for polynomials, the ERM converges in one or two iterations. That does not mean, however, that its computation time is better. It very much depends on the value of C, which I will discuss later.

In the case of convergence in the first iteration, the second plot is problematic.

2. The Townsend function often used for benchmarking optimization functions[1]

$$u(x, y) = -[\cos((x - 0.1)y)]^2 - x \sin(3x + y)$$

where:

$$X = [-2.25, 2.5]; Y = [-2.25, 1.75]$$

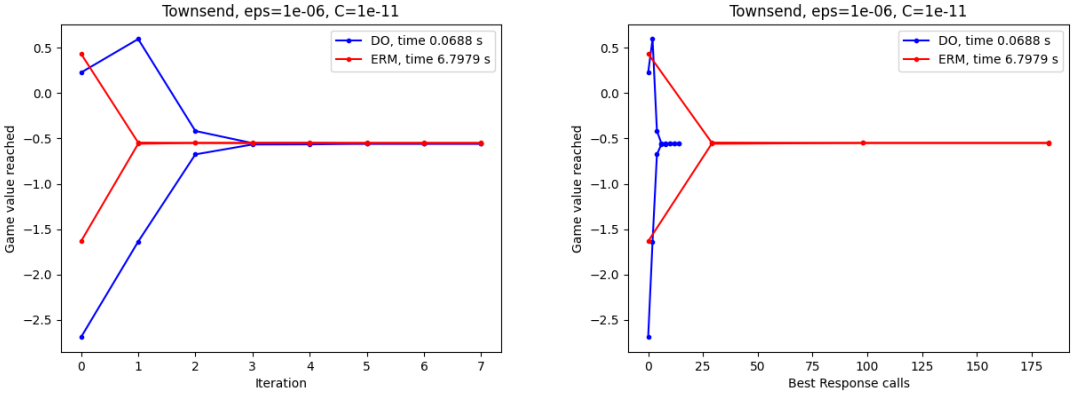

Figure 2.2. Converge comparison on the Townsend function

The trend is clear, ERM finds the ϵ-NE in a lower number of iterations, but each iteration requires computation of more best responses. There is a tradeoff.

[1] `https://en.wikipedia.org/wiki/Test_functions_for_optimization`

3. The Rosenbrock function often used for benchmarking optimization functions[1]

$$u(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

where:

$$X = Y = [-1.5, 1.5]$$

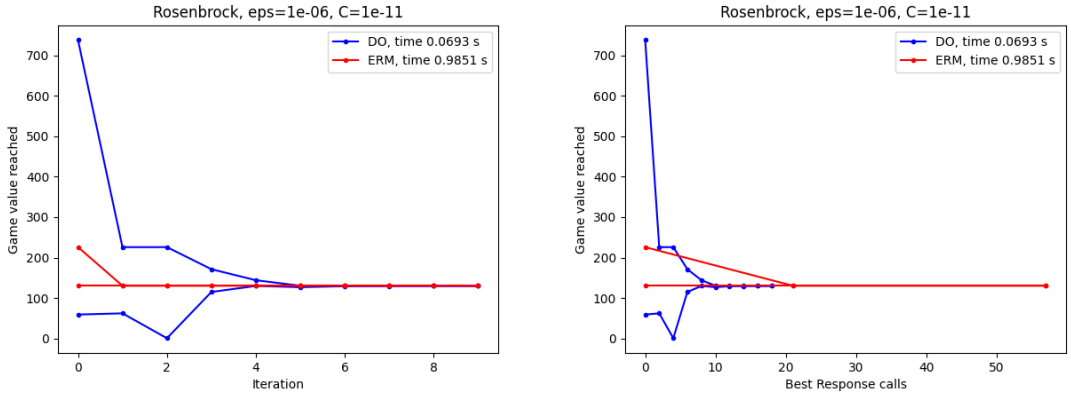

Figure 2.3. Converge comparison on the Rosenbrock function

4. Unknown function

$$u(x, y) = (1.5 - x + x \cdot y)^2 + (2.25 - x + x \cdot y^2)^2 + (2.625 - x + x \cdot y^3)^2$$

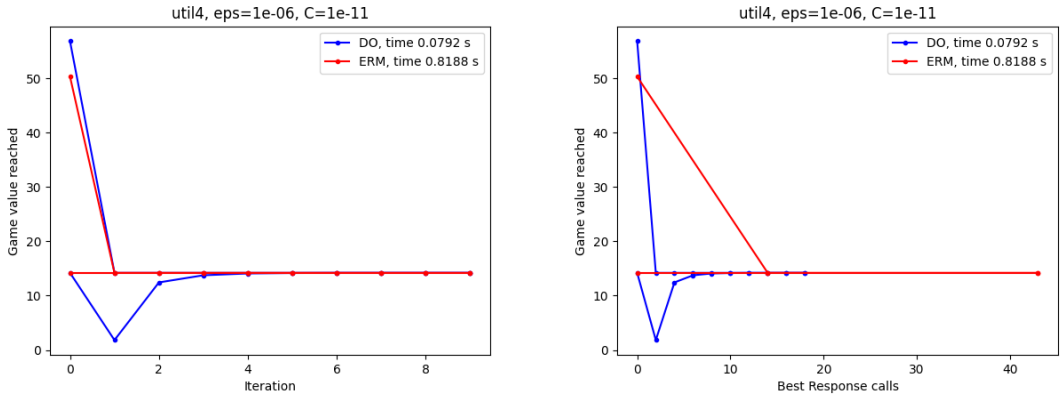

Figure 2.4. Converge comparison on the unknown function

where:

$$X = Y = [-1.5, 1.5]$$

[1] `https://en.wikipedia.org/wiki/Test_functions_for_optimization`

2.2.3 Convergence comparison, part 2: Functions taken from StayOnTheRidge research project

My colleague has compiled, in her research project[9], a list of functions. They are intended for experimenting with finding quasi-critical points, but since every game based on utility functions with hypercube domains has a $\epsilon-$NE, they can be used for this work, too. The compilation is available on GitLab[1]

Example 1
Example D from Appendix D[10][2]

$$u(x, y) = (x - 0.5) \cdot (y - 0.5)$$

where:

$$X = Y = [0, 1]$$

The solution presented in the paper: $(0.5, 0.5) \rightarrow u = 0$

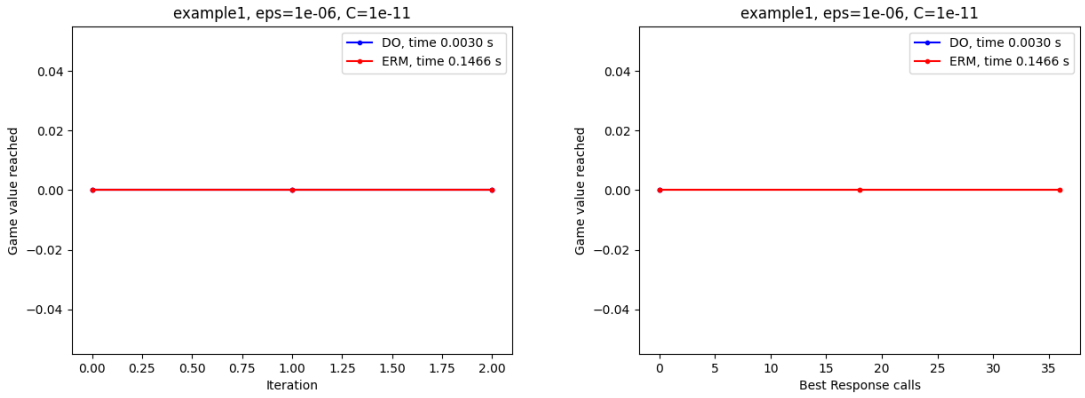

Figure 2.5. Converge comparison on Example 1

Both algorithms converged immediately. That is because the mixed NE is supported by the initial set of strategies, which are (as explained before) the bounds.

Example2 Is a finite two-player game and is not interesting for this experiment.

[1] https://gitlab.fel.cvut.cz/kosohmar/StayOnTheRidge.jl/-/blob/main/examples/examples.jl
[2] https://proceedings.mlr.press/v195/daskalakis23b/daskalakis23b.pdf

Example 3

Presented in Appendix E[10][1]

$$u(x,y) = -xy - \frac{1}{20}y^2 + \frac{2}{20} \cdot S(\frac{x^2+y^2}{2}) \cdot y^2$$

$$S(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 3z^2 - 2z^3 & \text{if } z \in (0,1) \\ 1 & \text{if } z \geq 1 \end{cases}$$

where:

$$X = Y = [-1, 1]$$

solution presented in the paper: $(0,0) \to u = 0$

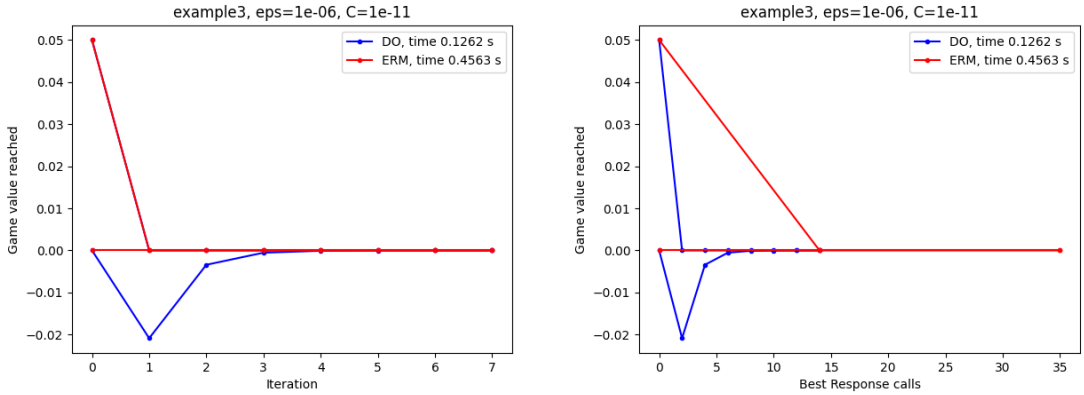

Figure 2.6. Converge comparison on Example 3

Example 4

Presented as the example 5.3 i) in the paper[11][2]

$$u(x,y) = p(x,y)/q(x,y)$$

$$p(x,y) = x_1^2 + x_2^2 - y_1^2 - y_2^2$$

$$q(x,y) = x_1 + y_1 + 1$$

where:

$$X = Y = [0,1]^2$$

solution presented in the paper:
$((0.2540, 0.2097) \cdot 10^{-4}, (0.2487, 0.2944) \cdot 10^{-4}) \to u = -4.00307647e - 10$

[1] https://proceedings.mlr.press/v195/daskalakis23b/daskalakis23b.pdf
[2] https://link.springer.com/article/10.1007/s10589-019-00141-6

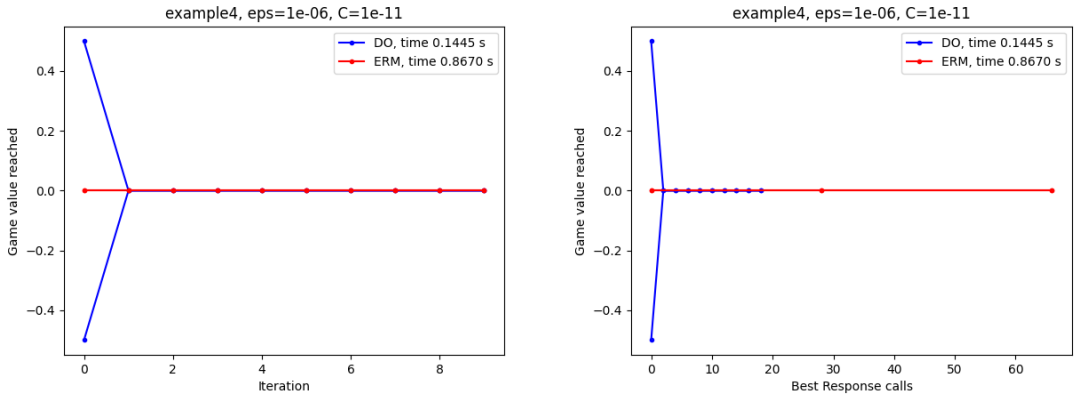

Figure 2.7. Converge comparison on Example 4

Example 5

Presented as the example 1 in the paper[12][1]

$$u(x, y) = 2xy^2 - x^2 - y$$

where:

$$X = Y = [-1, 1]$$

solution presented in the paper: $(0.4, 0.6) \rightarrow u = -0.47200000000000003$

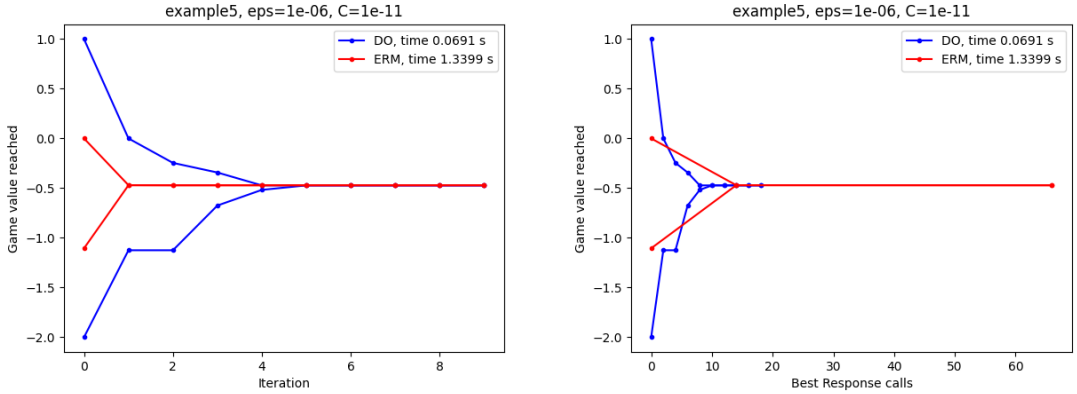

Figure 2.8. Converge comparison on Example 5

Example 6

Presented in Appendix E[10][2]

$$u(x, y) = (4x^2 - (y - 3x + \frac{x^3}{20})^2 - \frac{y^4}{10}) \exp(-\frac{x^2 + y^2}{100})$$

where:

$$X = Y = [-1, 1]$$

solution presented in the paper: $(0, 0) \rightarrow u = 0$

[1] https://arxiv.org/pdf/2109.04178.pdf
[2] https://proceedings.mlr.press/v195/daskalakis23b/daskalakis23b.pdf

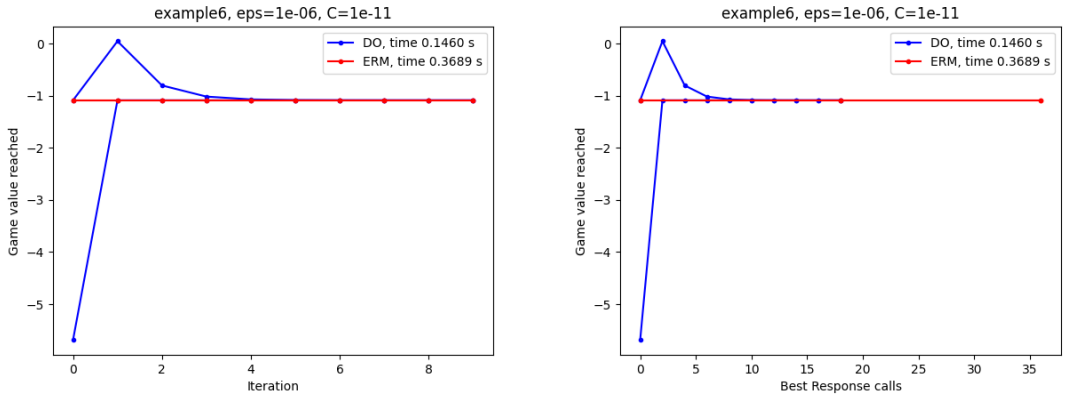

Figure 2.9. Converge comparison on Example 6, initialized on bounds

I have decided to also include the convergence plot for this game if the initial strategies are chosen randomly (instead of as the bounds). The convergence appears to be affected the most in this specific game.

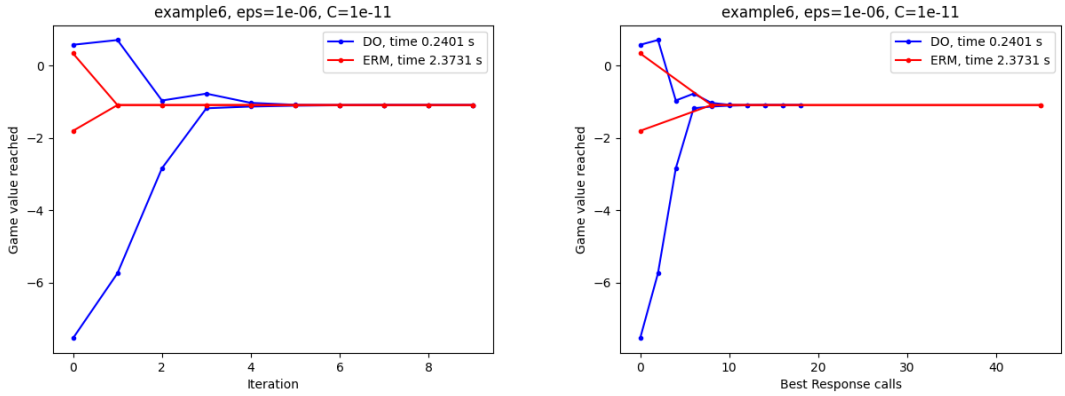

Figure 2.10. Converge comparison on Example 6, initialized randomly

While visually it is the Double Oracle's convergence, which is affected more, remember that the cost of every iteration of the ERM algorithm rises superlinearly, and its computation time increases significantly.

Example 7

Presented as the example 6.3 i) in the paper[13][1]

$$u(x,y) = \sum_{i=1}^{n}(x_i + y_i) + \sum_{i<j}(x_i^2 y_j^2 - y_i^2 x_j^2)$$

where:

$$X = Y = [-1,1]^3$$

solution presented in the paper: $(-1,-1,1),(1,1,1) \rightarrow u = 2$

[1] `https://proceedings.mlr.press/v195/daskalakis23b/daskalakis23b.pdf`

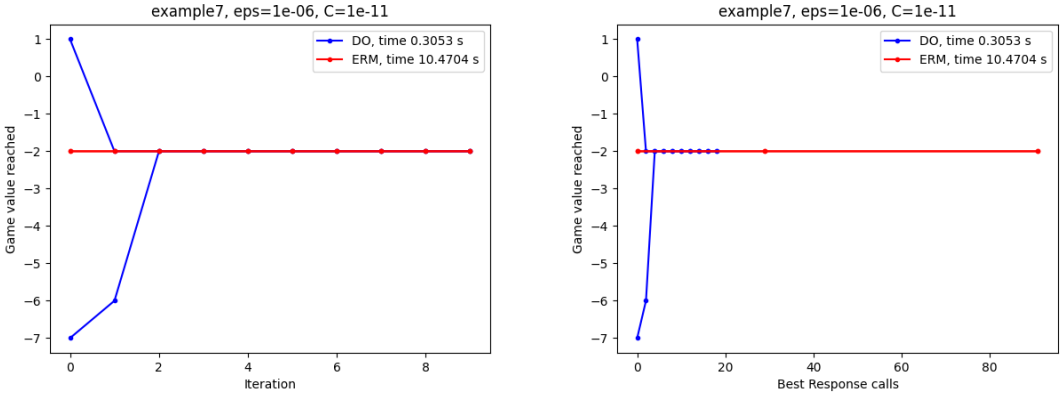

Figure 2.11. Converge comparison on Example 7

Example 8

Example 6.3 ii) from the paper[13][1]

$$u(x,y) = y^T y - x^T x + \sum_{1 \le i < j \le 3} (x_i y_j - x_j y_i)$$

where:

$$X = Y = [-1,1]^3$$

solution presented in the paper: $(-1,1,-1),(-1,1,-1) \to u = 1$

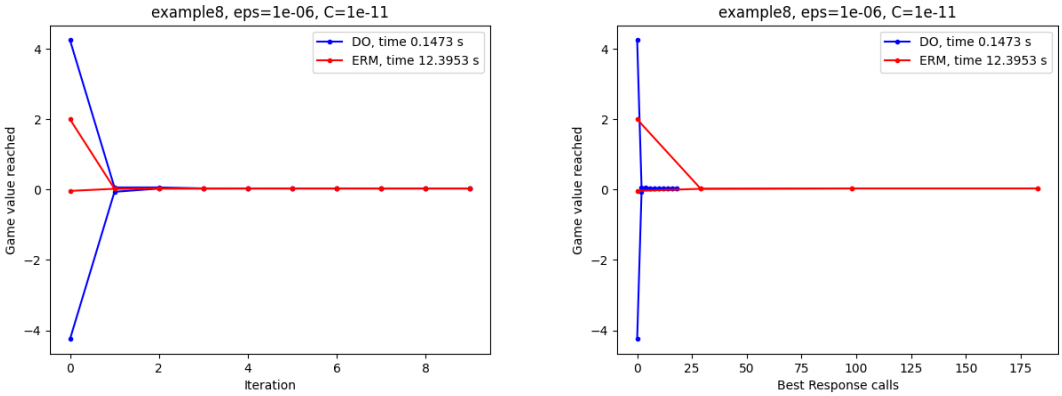

Figure 2.12. Converge comparison on Example 8

Example 9 $x^2 - y^2$

$$u(x,y) = x^2 - y^2$$

where:

$$X = Y = [-1,1]$$

well-known solution: $0,0 \to u = 0$

[1] `https://arxiv.org/pdf/1809.01218.pdf`

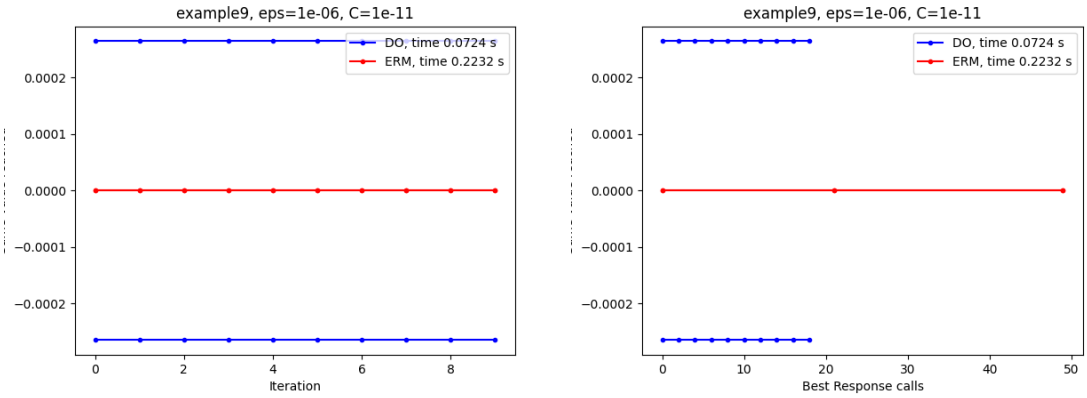

Figure 2.13. Converge comparison on Example 9

Example 10

The Monkey saddle

$$u(x, y) = x^3 - 3xy^2$$

where:

$$X = Y = [-1, 1]$$

well-known solution: $0, 0 \to u = 0$

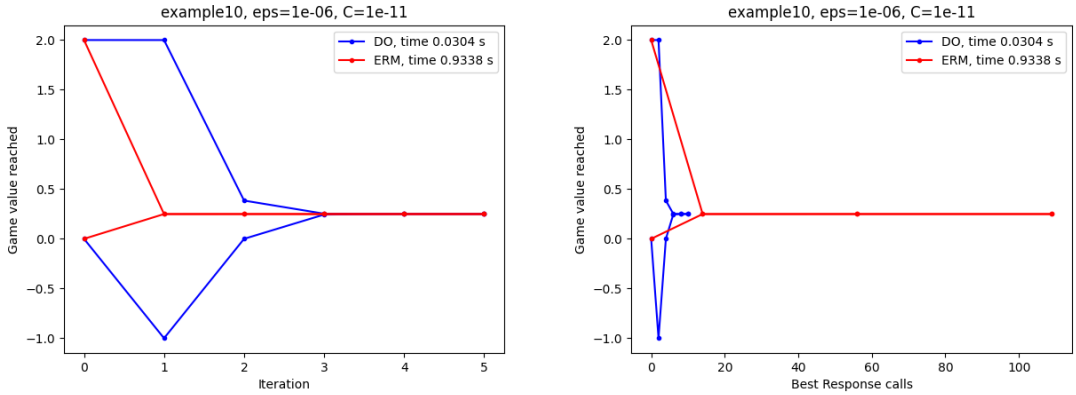

Figure 2.14. Converge comparison on Example 10

Example 11

example 1 from the paper[13][1]

$$u(x, y) = 0.2xy\cos(y)$$

where:

$$X = [-1, 1]; Y = [\pi, \pi]$$

solutions presented in the paper: $(0, -\pi), (0, \pi) \to u = -1$

[1] `https://arxiv.org/pdf/1809.01218.pdf`

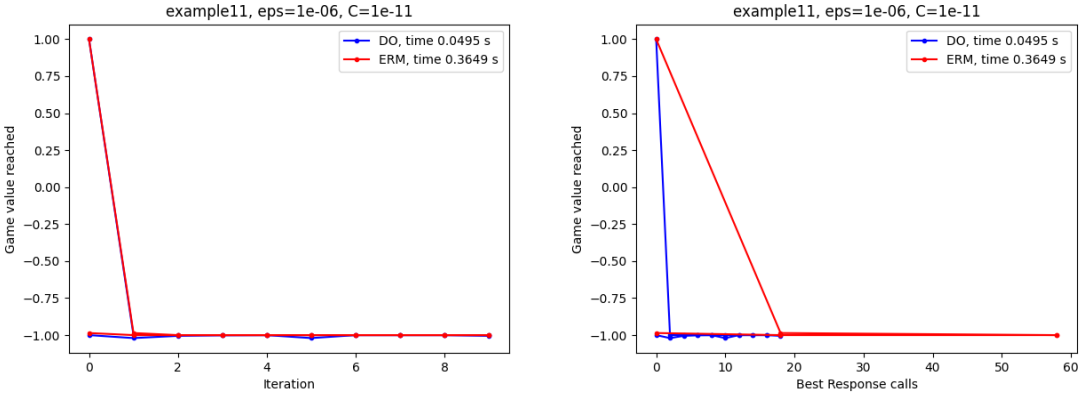

Figure 2.15. Converge comparison on Example 11

Example 12

Example 3 from the paper[13][1]

$$u(x.y) = x^3 + 2xy - y^2$$

where:

$$X = Y = [-1, 1]$$

solutions presented in the paper: $(0, 0), (-1, -1) \rightarrow u = 0$

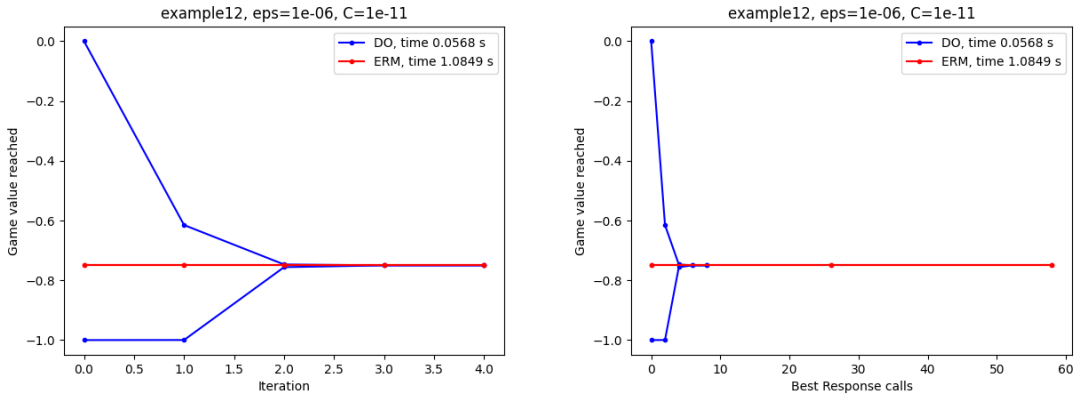

Figure 2.16. Converge comparison on Example 12

Example 13

Presented in figure 1 from the paper [14] [2]

$$u(x, y) = (x - \frac{1}{2}) * (y - \frac{1}{2}) + \frac{1}{3}\exp(-(x - \frac{1}{4})^2 - (y - \frac{3}{4})^2)$$

where:

$$X = Y = [-1, 1]$$

solution presented in the paper: $(0.4, 0.6) \rightarrow u = 0.3086658272776999$

[1] https://arxiv.org/pdf/1809.01218.pdf
[2] https://arxiv.org/pdf/1807.02629.pdf

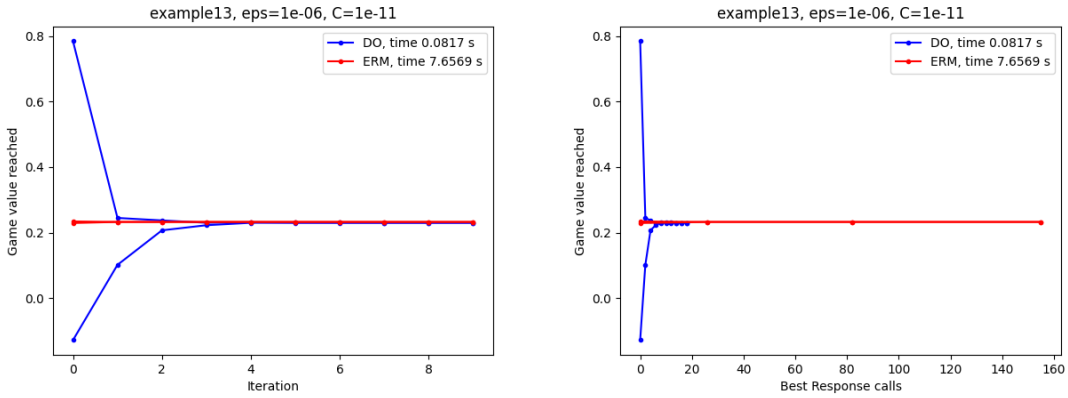

Figure 2.17. Converge comparison on Example 13

Example 14

Example 2.2 from paper [14] [1]

$$u(x, y) = (x^4 \cdot y^2 + x^2 + 1) \cdot (x^2 \cdot y^4 - y^2 + 1)$$

where:

$$X = Y = [-1, 1]$$

solutions presented in the paper: $(0, 0) \rightarrow u = 1$

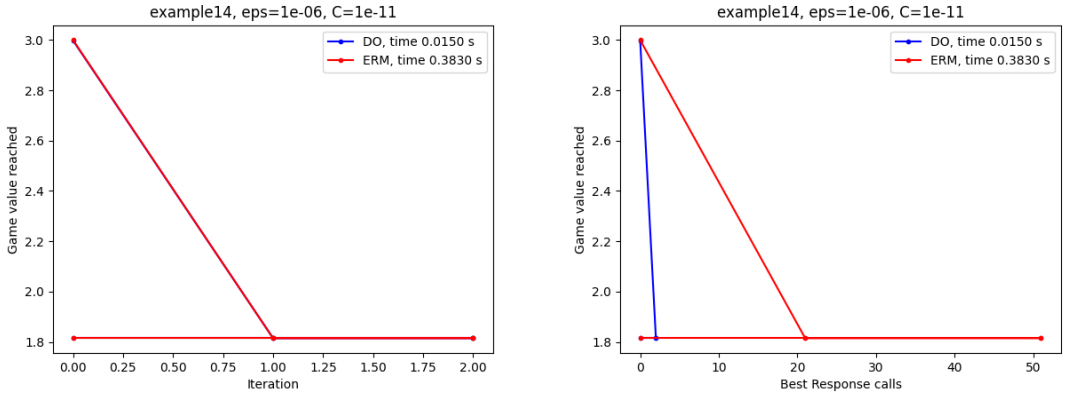

Figure 2.18. Converge comparison on Example 14

[1] `https://arxiv.org/pdf/1807.02629.pdf`

2.3 Testing claims regarding the ERM algorithm

The authors, in the paper proposing ERM for solving the ϵ-NE [1], have expressed multiple bounds for the computational complexity of the ERM algorithm. Here I present my experiments to hopefully support or disprove them. While the paper contains many of them, with varying importance, tightness, and calculation, in this project, I focus on these:

2.3.1 The first claim - convergence

Lemma C.2 (Nash for the half-infinite game)[1]: Let $G = (A = a_1, ..., a_n, Y, u)$ be a zero-sum game where $|A| = n$ and Y is possibly infinite, and let $\epsilon > 0$. Then, algorithm 3, executed with the parameter ϵ, finds an $O(\epsilon)$-Nash equilibrium, after $V = O(\log\ n/\epsilon^2)$ iterations.

The experiments

There are two parts to the claim about the sub-routine of the ERM algorithm. One is the bound on a number of steps to finish the computation. This obviously holds directly from the algorithm implementation, as $V \leftarrow \left\lceil \frac{C \log(n)}{\epsilon^2} \right\rceil$. Also, the bound does not mean much in practice, notice the $1/\epsilon^2$. Let's say you want to find a ϵ-NE, with $\epsilon = 10^{-6}$. Then this bound tells you it will take at most *approximately* 10^{12} iterations.

The second part is about the error of the computed. The unsettling issue is the dependence of the error on the chosen constant C, and on just on ϵ.

For this experiment, I have stuck to a well-known finite game of Rock-Paper-Scissors, defined by this matrix:

	0(=Rock)	1(=Paper)	2(=Scissors)
0 ‖	0	-1	1
1 ‖	1	-0	-1
2 ‖	-1	1	0

Table 2.1. Normal form of Rock-Paper-Scissors.

I have tried to find the NE by the ERM subroutine with different values of the magic constant C, $\epsilon = 10^{-6}$. The outcome is:

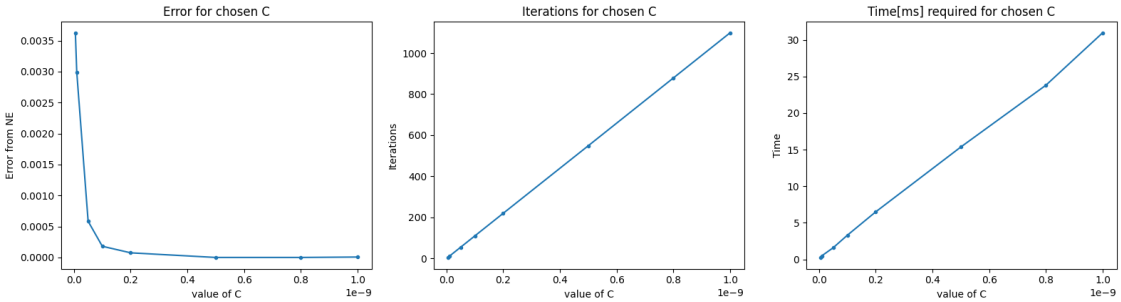

Table 2.2. Impact of value of C on the ERM performance, RPS

Notice that the error (difference between utility gained when compared to the real NE, which we know is 0 for this game) depends on C. That is because if C is not $suf ficiently$ large, there are not enough iterations in the MWU process, for the distribution of one player to settle.

The number of iterations (and time spent by the computation) increases linearly with the value of C, just as expected. The achieved error from the actual NE value decreases much faster than linearly, so it is possible to find a suitable C for every game, where both the error and t■e run time are reasonable. However, there is no direct way to get the right value of C before actually starting the computation, and by incorrectly setting the value of C, an arbitrary value of either error or run-time, can be obtained.

For a large (or a continuous) game, the situation can be even worse, as choosing a low value of C can lead to failing to identify the whole support of the requested ϵ-NE. Not only the number of *bestResponse* oracle calls must be at least the size of the support (of the ϵ-NE), but every single member of the support must become the *bestResponse* against the temporary mixed strategy of the opponent, so the required number of inner iterations, and consequently value of C, is not trivial.

To show the reach of the issue, I have remade the same experiment with the Rosenbrock function (which has infinite action spaces). As you can see, a wrong choice of C can lead to nonsense solutions.

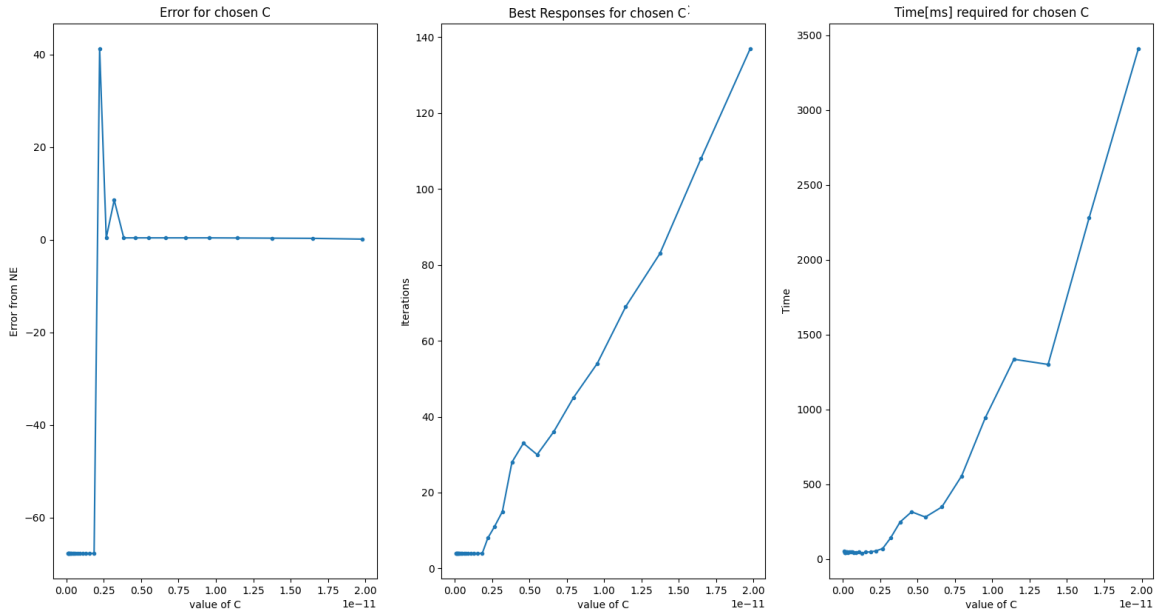

Table 2.3. Impact of value of C on the ERM performance, Rosenbrock function

The error shown is a difference between the value of the game in NE found by the Double Oracle and the Expected Regret Minimization. Remember the ERM algorithm finishes when the difference between computed upper and lower bounds of the temporary solution is at most ϵ, as the algorithm does not know the true value in NE. A the low value of C can cause the algorithm to get stuck in some subgame.

While it might seem that with the roughly 1/3 sec run time ERM reached a comparable outcome to the DO (which finished in 0.07 sec, finding a solution of 130.20), it is not necessarily true. The next plot shows the same experiment, but the value of C starts higher.

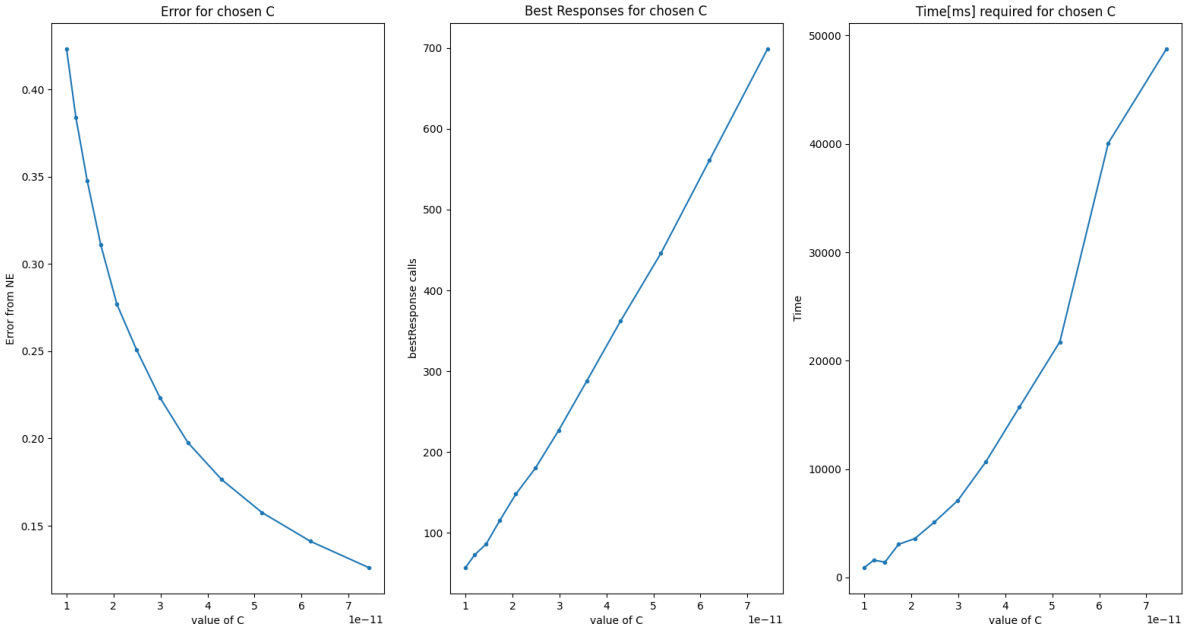

Table 2.4. Impact of value of C on the ERM performance, Rosenbrock function

At a certain point, increasing the value of C clearly leads to a closer convergence to the value found by Double Oracle. However, even after almost 1 minute of run time, the ERM algorithm managed to reach only an outcome roughly 0.1 % distant from the value found by DO.

The Rosenbrock function represents one of the more demanding functions in this set, and for most of the others, the ERM algorithm performs more convincingly.

2.3.2 The second claim - bounds for oracle calls

Lemma C.5 in [1]: Assume that algorithm 2 runs for T iterations. Then, the number of oracle calls is bounded by $O((T/\epsilon^2) \cdot (\log(T/\epsilon^2)))$

Experiments

The number of chosen specific actions per iteration is clearly bounded as claimed. That follows from the previous claim, in each iteration, $O(\log(T/\epsilon^2))$ more actions are added. Does this bound apply also for the number of oracle calls, as claimed?

I have tried to compute the NE of a game and just recorded the *bestResponse* (and action list sizes) counts. Notice the extreme values of both ϵ and C that had to be chosen.

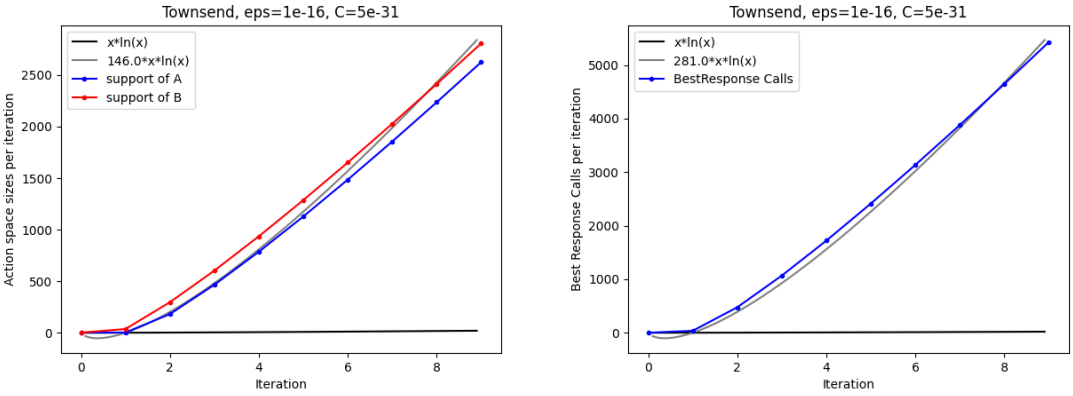

Figure 2.19. Computation complexity per iteration

As shown, both the considered action lists, as well as the *bestResponse* oracle calls can be asymptotically bounded by $n \cdot \ln(n) \cdot M$ for a suitable constant M. The claims hold, therefore the ERM algorithm is promising for certain use cases. One must take care when setting the value of C.

Chapter 3
Conclusion

3.1 Implications and possible future research.

It appears the ERM algorithm is worth using for solving the NE in this setting of zero-sum two-player games with continuous (hypercube) action spaces, at least in some use cases. As shown, it truly converges towards an NE. However, even though the ERM algorithm tends to converge to an NE in much a lower number of iterations than the DO algorithm, it requires more *bestResponse* oracle calls. And their computation, depending on the specific game, might not be cheap. Therefore there is a tradeoff between the number of iterations (and the LP computations, and the *bestResponse* computations. It then depends on the specific use case, and how does the computational complexity between the two oracles compare.

Also, the number of *bestResponse* computations is dependent on the value of the *magic constant* C, which is yet to be properly explained. The bigger the value of C, the higher the number of *bestResponse* calls, which might not be required for achieving a $\epsilon-$NE, but if C is too low, there is not enough space to explore the whole $\epsilon-$NE support, not all actions of an actual $\epsilon-$NE of the game are chosen, and the algorithm converges to a nonsense.

The approach to compute the adequate value of C is yet to be examined.

For this reason, the Double Oracle algorithm is probably the better option, when it comes to black box-like adoption, as its performance cannot be restricted by a poor parameter choice.

References

[1] Angelos Assos, Idan Attias, Yuval Dagan, Constantinos Daskalakis, and Maxwell Fishelson. *Online Learning and Solving Infinite Games with an ERM Oracle*. 12023.
https://proceedings.mlr.press/v195/assos23a/assos23a.pdf. In 36th Annual Conference on Learning Theory.

[2] Lukáš Adam, Rostislav Horčík, Tomáš Kasl, and Tomáš Kroupa. *Double Oracle Algorithm for Computing Equilibria in Continuous Games*. 2020.
https://arxiv.org/pdf/2009.12185.pdf. In: Vol. 35 No. 6: AAAI-21 Technical Tracks 6.

[3] Yoav Shoham, and Kevin Leyton-Brown. *Multiagent systems*. Cambridge University Press, 2008. ISBN 9780521899437.

[4] John Nash. *Non-Cooperative Games*. September 1951.
https://www.cs.vu.nl/~eliens/download/paper-Nash51.pdf. In: Annals of Mathematics, Second Series, Vol. 54, N^o. 2 , pages 286-295 .

[5] George W. Brown. *Iterative Solutions of Games by Fictitious Play*. 1951.
https://www.math.ucla.edu/~tom/stat596/fictitious.pdf. In Activity Analysis of Production and Allocation.

[6] Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. *A Double Oracle Algorithm for Zero-Sum Security Games on Graphs*. 2011.
http://www.cs.cmu.edu/~conitzer/graph_securityAAMAS11.pdf. In: The 10th International Conference on Autonomous Agents and Multiagent Systems Volume 1, Pages 327–334.

[7] Vincent Conitzer. *Approximation Guarantees for Fictitious Play*. 2009.
http://www.cs.cmu.edu/~conitzer/fictitiousAllerton09.pdf. In Proceedings of the 47th annual Allerton conference on Communication, control, and computing, Pages 636–643.

[8] M. J. D. Powell. *An efficient method for finding the minimum of a function of several variables without calculating derivatives*. 1964.
https://academic.oup.com/comjnl/article/7/2/155/335330. In: The Computer Journal, Volume 7, Issue 2, 1964, Pages 155–162.

[9] Marika Kosohorská. *Bachelor software or research project on FEE CTU*. 2024.
https://gitlab.fel.cvut.cz/kosohmar/StayOnTheRidge.jl/-/blob/main/examples/examples.jl.

[10] Constantinos Daskalakis, Noah Golowich, Stratis Skoulakis, and Manolis Zampetakis. *STay-ON-the-Ridge: Guaranteed Convergence to Min-Max Critical Points in Nonconvex-Nonconcave Games*. 2023.
https://proceedings.mlr.press/v195/daskalakis23b/daskalakis23b.pdf. In: 36th Annual Conference on Learning Theory.

[11] Guangming Zhou, Qin Wang, and Wenjie Zhao. *Saddle points of rational functions*. 2020.
https://link.springer.com/article/10.1007/s10589-019-00141-6. In: Computational Optimization and Applications, Volume 75, pages 817–832.

[12] Tomáš Kroupa, and Tomáš Votroubek. *Multiple Oracle Algorithm to Solve Continuous Games*. 2022.
https://arxiv.org/pdf/1809.01218.pdf. In: GameSec 2022: Decision and Game Theory for Security pp 149–167.

[13] Jiawang Nie, Zi Yang, and Guangming Zhou. *The Saddle Point Problem of Polynomials*. 2021.
https://arxiv.org/pdf/1809.01218.pdf. In: Foundations of Computational Mathematics, Volume 22, pages 1133–1169.

[14] Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. *Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile*. 2020.
https://arxiv.org/pdf/1809.01218.pdf. In: International Conference on Learning Representations 2018.

Appendix A
Assignement

Můj projekt

B231

Stav:	Téma bylo úspěšně schválené. Můžete začít pracovat na projektu.
Název tématu:	Řešení spojitých her pomocí orákl algoritmů
Vedoucí:	doc. Ing. Tomáš Kroupa, Ph.D.
Určeno pro:	Magisterské studium
Obsazenost:	1 / 1
Katedra vedoucího:	Katedra počítačů
Popis:	Řešení spojitých her s kompaktními prostory strategií představuje velkou výzvu pro současnou teorii her. Hledání smíšených ekvilibrií ve dvouhráčových modelech s nulovým součtem je obtížně řešitelný problém, k jehož problému se často používají numerické iterační metody založené na různých oracle přístupech (výpočet best response atd.) Nový článek [2] je založen na jednom takovém přístupu, jehož varianta byla již kdysi použita v práci [1].
Požadavky:	1. Seznámit se s metodami online learning pro řešení her v normální formě 2. Prostudovat článek [2] s důrazem na pochopení Algorithm 3. 3. Implementovat tento algoritmus a porovnat jeho výsledky s Double Oracle v článku [1].
Literatura:	[1] L. Adam, R. Horčík, T. Kasl, T. Kroupa. Double Oracle Algorithm for Computing Equilibria in Continuous Games. Proceedings of AAAI 2021. [2] Assos, Angelos, et al. "Online learning and solving infinite games with an erm oracle." The Thirty Sixth Annual Conference on Learning Theory. PMLR, 2023.
Studijní programy:	